Study on Stock Trading and Portfolio Optimization using Genetic Network Programming

CHEN, Yan

February 2010

Waseda University Doctoral Dissertation

Study on Stock Trading and Portfolio Optimization using Genetic Network Programming

CHEN, Yan

Graduate School of Information, Production and Systems

Waseda University

February 2010

Abstract

Research on stock price prediction and trading model using evolutionary computation has been done in recent years. As we know, prediction in the stock market is quite difficult for a number of reasons. First, the ultimate goal of our research is not to minimize the prediction error, but to maximize the profits. Second, the weak relationships among variables tend to be nonlinear, and they may hold only in limited areas of the search space. Finally, since the stock market data are given in an event-driven way, they are highly influenced by the indeterminate dealing. Generally speaking, there are two kinds of methods for predicting stock prices and determining the timing of buying or selling stocks: one is fundamental analysis which analyzes stock prices using the financial statement of each company, the economic trend and movements of the exchange rate; the other is technical analysis which analyzes numerically the past movement of stock prices. The proposed method belongs to technical analysis since it determines the stock trading actions based on the technical indices such as Relative Strength Index, MACD, Golden/Dead Cross and so on.

The most recent literature in the related fields exposed Portfolio Optimization, Investment Strategy Determination, and Market Risk Analysis as three major trends in the utilization of evolutionary algorithms. Our research focuses on the problem of Investment Strategy Determination and Portfolio Optimization through the use of Genetic Network Programming (GNP) with reinforcement learning technique. The objective of this work is to provide an unique technique of decision-making for investors. First, it presents a stock trading model based on GNP and Sarsa learning by the use of Importance Index (IMX) and candlestick charts. Appropriate trading actions can be determined with the proposed model depending on the situation. Second, it extends an application of GNP with control node (GNPcn) to the portfolio optimization problem. Third, it proposes a new method name Genetic Relation Algorithm (GRA) and applies

it to the large-scale portfolio selection problem.

Three advances were made in the study including the efficient stock trading rules, multi brands optimization and portfolio selection. The main realized results are presented as below.

1. Efficient stock trading rules using Genetic Network Programming with reinforcement learning

It has been clarified that GNP is an effective method mainly for dynamic problems since GNP represents its solutions using graph structures, which contributes to creating quite compact programs and implicitly memorizing past action sequences in the network flows. When GNP is combined with Sarsa learning, an effective stock trading model can be made and it has two advantages: one of them is online learning, and another advantage is the combination of a diversified search of GNP and an intensified search of Sarsa. Moreover, Importance Index (IMX) and candlestick charts are introduced for stock trading decision making. The simulation results show that GNP-Sarsa has obtained good profits and outperforms many other tradition methods in the evolutionary computation field.

Furthermore, in order to improve the performance of GNP-Sarsa algorithm, we develop an enhanced stock trading model for adapting to the change of stock prices, which is called Real Time Updating Genetic Network Programming (RTU-GNP). Compared with GNP-Sarsa, the RTU-GNP method makes a stock trading decision considering both the recommendable information of technical indices and candlestick charts according to the real time stock prices, in which sliding window is introduced to the model. Also, as another new point, we propose a method that can learn the appropriate function describing the relation between each technical index and the IMX. The experimental results show that RTU-GNP can get more profits than traditional methods, while the period of sliding window is set as 7 days.

2. A portfolio optimization model using Genetic Network Programming with control nodes

This is an extended method of GNP-Sarsa, which can deal with multi brands in a portfolio simultaneously. Since GNP has a directed graph structure, GNPcn has improved the performance of GNP by extending the evolutionary method of it, i.e., the breadth

and depth of searching space for GNP. The main problem that we have solved in this part is how to allocate the available capital to different stock brands in order to maximize the profit. The efficiency of GNPcn model has been confirmed by the experimental results.

3. A portfolio selection model using Genetic Relation Algorithm and Genetic Network Programming

The third part presents a new approach named Genetic Relation Algorithm (GRA), which is designed for the portfolio selection. Given the investor's objectives and economic conditions, we can find out what stock brands to include in an optimal portfolio by using the proposed GRA method, and thus maximize the expected return and minimize risk simultaneously. In order to pick up the most efficient portfolio from a large number of brands, GRA considers the correlation coefficient between stock brands as strength, which indicates the relation between nodes in GRA individuals. The algorithm evaluates the relationships between stock brands using a specific measure of strength and generates the optimal portfolio in the final generation. The performance of the portfolio selected with GRA is confirmed by the GNPcn stock trading model. Moreover, a new genetic operator named guided mutation has been developed in the final part, with which the resultant solution can hopefully fall in or close to a promising area. Therefore, the exploitation ability of evolution has been improved. By carrying out the simulations, GRA with guided mutation (GRA/G) shows the effectiveness for portfolio selection.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Evolutionary Computation

Evolutionary Computation (EC) [1]-[13] is well-known for producing the solutions in optimization problems based on change, composition and selection. Since EC techniques can deal with complex optimization problems better than traditional techniques, they have attracted increasing attentions in recent years for solving the optimization problems. They are more robust than traditional methods based on formal logics or mathematical programming for many real world applications. In this research, EC has been found particularly useful in the financial field.

As we know, many optimization problems from economic and industrial world, in particular the financial problems, are very complex in nature and quite hard to solve by conventional techniques. Since 1960s, there has been an increasing interest in imitating living beings to solve such kinds of hard problems. Simulating natural evolutionary process of human beings results in stochastic optimization techniques called Evolutionary Computation that can often outperform conventional methods when applied to difficult real world problems. There are currently four main avenues of this research: Genetic Algorithm (GA) [1], [2], Evolutionary Programming (EP) [3], Evolution Strategies (ESs) [4], [5] and Genetic Programming (GP) [6], [7]. Among them, GA and GP are perhaps the most widely known types of Evolutionary Computation today. And the proposed method in our research is an extension of GA and GP.

When Evolutionary Computation techniques are applied to the real world, several EC components should be considered. Firstly, we decide the genetic representation of solution. Secondly, we define the fitness evaluation of the solution using the objective

functions. Lastly, genetic operators such as crossover operator, mutation operator and selection methods are applied to the population of EC. These implementation processes are repeated until the predefined generation number is satisfied or the optimal solution is reached.

EC has received considerable attention regarding their potential as a novel optimization technique. There are three major advantages when applying EC to optimization problems:

1. Adaptability: EC does not have much mathematical requirements about the optimization problems. Due to the evolutionary nature, EC will search for solutions without regard to the specific inner workings of the problem. EC can handle any kind of objective functions and any kind of constraints, i.e., linear or nonlinear, defined on discrete, continuous or mixed search spaces.

2. Robustness: The use of evolution operators makes EC very effective in performing global search, while most of conventional heuristics usually perform local search. It has been proved by many studies that EC is more efficient and more robust in locating optimal solution and reducing computational effort than other conventional heuristics.

3. Flexibility: EC provides us a great flexibility to hybridize with domain-dependent heuristics to make an efficient implementation for a specific problem.

1.2 Motivation and Objective

Prediction in financial domains, especially in stock market is quite difficult for a number of reasons. First, the ultimate goal of our research is not to minimize the prediction error, but to maximize the profits. It forces us to consider a large number of independent variables, thereby increasing the dimensionality of the search space. Second, the weak relationships among variables tend to be nonlinear, and may hold only in limited areas of the search space. Especially, the data in stock markets are highly time-variant and changing every minute. Third, the stock market data are given in an event-driven way. They are highly influenced by the indeterminate dealing. In financial practice, the key is to find the hidden interactions among variables.

Traditional investment activities are based on technical analysis, market sentiment (asymmetric information, rumors, noise trading) and imitative behaviors. This leads to unjustified bias in decision making. To remove such subjectivity, in this thesis, we try to provide an integrated intelligent model for the investors to decide whether to buy or sell the stocks and how to select the stocks in a portfolio. The objective of this work is to provide two main investment strategies: one is for stock trading based on technical analysis, and the other is for portfolio optimization.

To facilitate this objective, we construct the stock trading model and portfolio optimization systems using Genetic Network Programming (GNP) [14]-[19] with reinforcement learning [26]-[32] for stock brands that are trained with data from Japan's stock market.

1.3 Structure of This Thesis

This thesis falls naturally into three parts, which are relatively independent:

- Study of trading rules on stock markets using Genetic Network Programming with reinforcement learning (Chapters 2 and 3).
- A portfolio optimization model using Genetic Network Programming with control nodes (Chapter 4).
- A Portfolio Selection Model using Genetic Relation Algorithm and Genetic Network Programming (Chapter 5 and 6).

The first two parts form the main body of the thesis, and they are both devoted to the study of stock trading systems. The first part probably contains the most significant experimental results; this is reflected in the choice of thesis title. The second part can be viewed as a description of portfolio optimization model that play a supporting role in the first part. However, there will also turn out to be a surprising reciprocal connection, namely that results from the first part will provide a much-needed explanation for the success of a very efficient algorithm in the second part. The intertwining of these two subject areas had turned out to be one of the most beautiful surprises in this body of research.

The third part forms essentially separate topic, and can therefore be read independently. However they do share with the rest of the thesis the common theme of investment strategy: the third part presents a new approach to the portfolio selection, which can be applied to the stock trading model described in the first two parts.

The goals and subject matter of the three parts are sufficiently different with each other, which now follow the individual introductions as below.

1.3.1 Chapters 2 and 3: Study of trading rules on stock markets using Genetic Network Programming with reinforcement learning

We have proposed Genetic Network Programming (GNP) [14]-[19] as an extended method of Genetic Algorithm [1], [2] and Genetic Programming [6]-[25]. It has been clarified that GNP is an effective method mainly for dynamic problems since GNP represents its solutions using graph structures, which contributes to creating quite compact programs and implicitly memorizing past action sequences in the network flows.

In Chapter 2 we propose an extended algorithm of GNP which combines evolution and reinforcement learning [26]-[32], i.e., Genetic Network Programming with Sarsa Learning (GNP-Sarsa). GNP-Sarsa has two advantages: one of them is online learning, and another advantage is the combination of a diversified search of GNP and an intensified search of Sarsa. The detailed explanation is presented in the latter parts. Generally speaking, there are three important points in the proposed method. First, we combine GNP and Sarsa Learning which is one of the reinforcement learning methods, while Importance Index (IMX) and candlestick charts are introduced for efficient stock trading decision making. Second, although there are so many technical indices in the technical analysis, GNP with Sarsa can select appropriate indices and candlestick charts to judge the buying and selling timing of stocks. The third point is that subnodes are introduced in each node to determine appropriate trading actions and select stock price information depending on the situation.

In Chapter 3 we introduce an enhanced stock trading model based on the GNP-Sarsa algorithm for adapting to the change of stock prices, which is called Real Time Updating Genetic Network Programming (RTU-GNP). Compared with GNP-Sarsa, the RTU-GNP method makes a stock trading decision considering both the recommendable information of technical indices and candlestick charts according to the real time

stock prices, in which sliding window is introduced to the model. Moreover, in order to improve the performances of the previous GNP-Sarsa algorithm, we propose a new method in Chapter 3 that can learn the appropriate function describing the relation between each technical index and the IMX. This is an important point that devotes to the enhancement of the traditional algorithm.

1.3.2 Chapter 4: A portfolio optimization model using Genetic Network Programming with control nodes

Since GNP-Sarsa stock trading model is introduced in the previous part, which can generate trading rules for individual stock brand efficiently, an extended method is needed to deal with multi-brands in a portfolio simultaneously.

In order to extend the functions of conventional GNP, Genetic Network Programming with control nodes (GNPcn) is developed. Since GNP has a directed graph structure, the aim of GNPcn is to improve the performance of GNP by extending the evolutionary method of it, i.e., the breadth and depth of searching space for GNP.

In Chapter 4 an application of GNPcn to the portfolio optimization problem is studied. Portfolio optimization in the stock market consists of deciding what brands to include in a portfolio given the investor's objectives and economic conditions. The always difficult selection process includes identifying which brands to purchase, how much, and when. A rational investor needs to consider not only maximizing the profit of the investment, but also minimizing the uncertainty or risk resulting from the fluctuations that are expected in the value of the portfolio. The main problem in this chapter is how to allocate the available capital to different stock brands in order to maximize the profit.

1.3.3 Chapter 5 and 6: A portfolio selection model using Genetic Relation Algorithm and Genetic Network Programming

The final part forms essentially separate topic, and can therefore be read independently. However it shares the common theme of investment strategy with the rest of the thesis: the third part presents a new approach named Genetic Relation Algorithm (GRA), which is designed for the portfolio selection especially.

In Chapter 5 we present an application of the evolutionary computation method named Genetic Relation Algorithm (GRA) to the problem of portfolio selection. In

the conventional portfolio optimization problems, given the investor's objectives and economic conditions, we want to find out what assets to include in an optimal portfolio, in order to maximize the expected return and minimize risk simultaneously. Since the number of brands in the stock market is generally very large, techniques for selecting the effective portfolio are likely to be of interest in the financial field.

Due to the bottlenecks of other traditional Artificial Intelligence (AI) approaches, GRA is developed and firstly applied to the portfolio selection problem. In order to pick up the most efficient portfolio from a large number of brands, the proposed model considers the correlation coefficient between stock brands as strength, which indicates the relation between nodes in GRA. The algorithm evaluates the relationships between stock brands using a specific measure of strength and generates the optimal portfolio in the final generation. The performance of the portfolio selected with GRA is confirmed by the GNPcn stock trading model. Generally speaking, the contributions of the proposed method are as follows: first, in the conventional Markowitz methods, a combination of brands is given, then the distribution ratio of the capital to each brand is determined by considering the return and risk. However, in the proposed GRA method, the best combination of stocks is selected from a large number of brands by considering the correlation coefficients of the brands. Then, the distribution of initial capital and the trading strategy are determined by GNPcn. Second, the number of stock brands in the best portfolio can be flexibly defined by users since the brands correspond to nodes in the GRA individuals.

In order the improve the performance of GRA portfolio selection model, Chapter 6 takes a different approach by applying Genetic Relation Algorithm with guided mutation (GRA/G) and GNPcn to construct a portfolio selection and stock trading model. In a sense, the proposed model is an integrated intelligent model. Especially, guided mutation can be regarded as an improvement of the conventional mutation operators, with which the resultant solution can hopefully fall in or close to a promising area. In such a way, the similarity between an offspring and its parent can be controlled to some extent. The main feature of GRA/G is that, guided mutation generates offspring according to the average value of correlation coefficients in each GRA individual, which means to enhance the exploitation ability of evolution.

Chapter 2

Trading Rules on Stock Markets Using Genetic Network Programming with Sarsa Learning

2.1 Introduction

Evolutionary Computation is well-known for producing the solutions in optimization problems based on change, composition and selection. We have proposed Genetic Network Programming (GNP) [17]-[19] as an extended method of Genetic Algorithm (GA) [1], [2] and Genetic Programming (GP) [6], [7]. It has been clarified that GNP is an effective method mainly for dynamic problems since GNP represents its solutions using graph structures, which contributes to creating quite compact programs and implicitly memorizing past action sequences in the network flows. Moreover, we proposed an extended algorithm of GNP which combines evolution and reinforcement learning [26] (GNP-RL). GNP-RL has two advantages, and one of them is online learning. Since original GNP is based on evolution only, the programs are evolved mainly after task execution or enough trial, i.e., offline learning. On the other hand, the programs in GNP-RL can be changed incrementally based on rewards obtained during task execution, i.e., online learning. Concretely speaking, when an agent takes a good action with a positive reward at a certain state, the action is reinforced and when visiting the state again, the same action will be adopted with higher probability. Another advantage of GNP-RL is the combination of a diversified search of GNP and an intensified search of RL. The role of evolution is to make rough structures through selection, crossover and mutation, while the role of RL is to determine one appropriate path in a structure

made by evolution. Diversified search of evolution could change programs largely with which the programs could escape from local minima. RL is executed based on immediate rewards obtained after taking actions, therefore intensified search can be executed efficiently.

Research on stock price prediction and trading model using evolutionary computation and neural networks has been done [33]-[35] in recent years. Generally speaking, there are two kinds of methods for predicting stock prices and determining the timing of buying or selling stocks: one is fundamental analysis which analyzes stock prices using the financial statement of each company, the economic trend and movements of the exchange rate; the other is technical analysis [36] which analyzes numerically the past movement of stock prices. The proposed method belongs to technical analysis since it determines the timing of buying and selling stocks based on the technical indices such as Relative Strength Index, MACD, Golden/Dead Cross and so on.

There are three important points in this chapter. First, we combine GNP and Sarsa Learning [37] which is one of the reinforcement learning methods, while Importance Index (IMX) and candlestick charts [38]-[41] are introduced for efficient stock trading decision making. Concretely speaking, Sarsa is used to select appropriate actions (buying/selling), stock price information obtained from IMX and candlestick charts through the experiences during the trading. IMX and candlestick charts tell GNP whether or not the buying or selling signals are likely to appear at the current day. Second, although there are so many technical indices in the technical analysis, GNP with Sarsa can select appropriate indices and also select candlestick charts to judge the buying and selling timing of stocks. In other words, GNP with Sarsa could optimize the combinations of the information obtained by technical indices and candlestick charts. The third important point is that sub-nodes are introduced in each node to determine appropriate actions (buying/selling) and to select appropriate stock price information depending on the situation.

This chapter is organized as follows: In Section 2.2, the related works are described. In Section 2.3, the algorithm of the proposed method is described. Section 2.4 shows simulation environments, conditions and results. Section 2.5 is devoted to conclusions.

2.2 Related Works

Prediction in financial domains, especially in stock market is quite difficult for a number of reasons. First, the ultimate goal of our research is not to minimize the prediction error, but to maximize the profits. It forces us to consider a large number of independent variables, thereby increasing the dimensionality of the search space. Second, the weak relationships among variables tend to be nonlinear, and may hold only in limited areas of the search space. Especially, the data in stock markets are highly time-variant and changing every minute. Third, the stock market data are given in an event-driven way. They are highly influenced by the indeterminate dealing. In financial practice, the key is to find the hidden interactions among variables [42].

Stock market analysis has been one of the most actively pursued avenues of Machine Learning (ML) research and applications. The most recent literature in the related fields exposed Portfolio Optimization, Investment Strategy Determination, and Market Risk Analysis as three major trends in the utilization of Machine Learning approaches. Portfolio Optimization focuses on the correlative properties of stock market data in order to extract mutual dependency (or independency) information [43]-[45]. Investment Strategy Determination addresses financial prediction based on financial index analysis for the purposes of investment decision-making. Various Neural Network approaches are by far the most commonly taken route in the related works. However, other alternative methods exist, such as Support Vector Machines [46], Genetic Algorithms [47] and statistical analysis [48]. The Market Risk Analysis concentrates on the evaluation of the risk factors involved in various investment options, such as expected return and volatility. An example of an overall market risk evaluation system is described in [49]. Our research focuses on the problem of Investment Strategy Determination through the use of GNP with reinforcement learning technique.

In recent years, evolutionary algorithms have been applied to several financial problems. There have been several applications of GA to the financial problems, such as portfolio optimization [50], bankruptcy prediction [51], financial forecasting [52]-[54], fraud detection and scheduling [50], [57], [58]. GP has also been applied to many problems in the time-series prediction and trading system [55], [56].

In our research, we propose Genetic Network Programming with Sarsa Learning

for creating trading rules on stock markets. GNP has the following advantages in the financial prediction field. First, GNP has a memory function because of its graph structure, i.e., judgment nodes and processing node are connected to each other in a network. As stock markets are highly influenced by the time, we can consider the information in the past well by the memory function of GNP for creating the effective programs. Second, GNP works extremely well for dealing with the stock market problems. That is because GNP has a quite compact structure and it can reuse the nodes for many times. By using GNP we can create effective trading rules in the stock market, and we can also save the calculation time and memory consumption because of the compact structures of GNP. By combining GNP with Sarsa Learning in this chapter, we get more advantages such as the combination of online learning and offline learning, diversified search and intensified search.

2.3 GNP with Sarsa (GNP-Sarsa) and Its Trading Algorithm

2.3.1 Basic Structure of GNP-Sarsa

Fig. 2.1 shows a basic structure of GNP-Sarsa and Fig. 2.2 shows judgment node and processing node structures. GNP-Sarsa consists of judgment nodes and processing nodes, which are connected to each other. Judgment nodes have if-then type branch decision functions. They return judgment results for assigned inputs and determine the next node. Processing nodes take actions (buying or selling stocks). While judgment nodes have conditional branches, processing nodes have no conditional branches. The role of a start node is to determine the first node to be executed. The graph structure of GNP has some inherent characteristics such as compact structures and an implicit memory function that contributes to creating effective action rules as described in section 2.2. GNP-Sarsa has two kinds of time delays: time delays GNP-Sarsa spend on judgment or processing, and the ones it spends on node transitions. In this chapter, the role of time delays is to determine the maximum number of technical indices and candlestick information to be considered when GNP-Sarsa determines the buying or selling at a certain day.

In the table of node gene, K_i represents the node type, K_i=0 means start node, K_i=1 means judgment node and K_i=2 means processing node. ID_i represents an identifica-

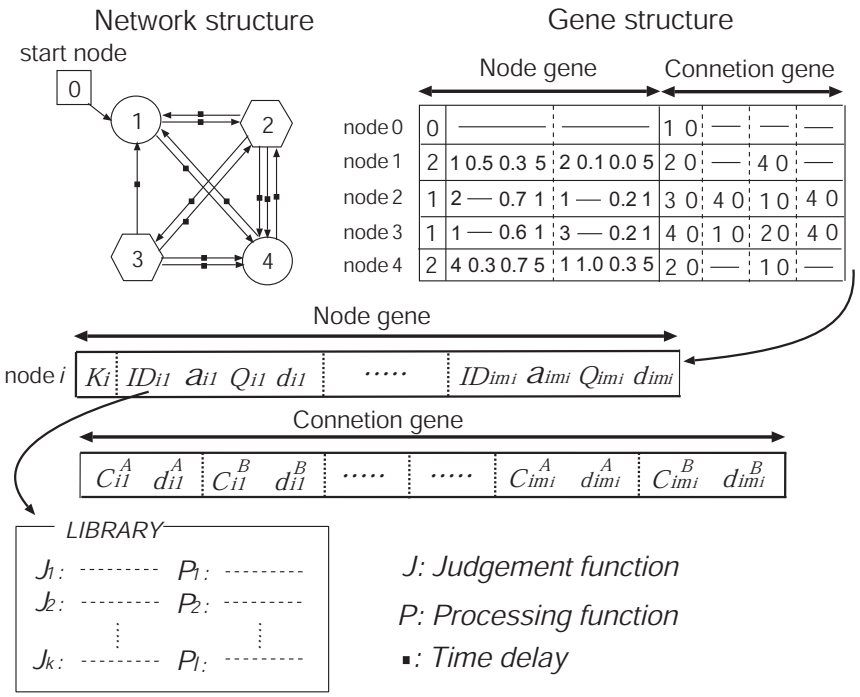


Figure 2.1: Basic structure of GNP with Sarsa

tion number of the node function, e.g., K_i=1 and ID_i=2 mean the node is J_2. a_{ip} is a parameter which represents the threshold for determining buying or selling stocks in a processing node. Q_{ip} means Q value which is assigned to each state and action pair. In this method, "state" means a current node, and "action" means a selection of a sub-node (node function). In general reinforcement learning framework, the current state is determined by the combination of the current information, and action is an actual action an agent takes, e.g., buying or selling stocks. However, in GNP-Sarsa, the current node is defined as the current state, and a selection of a sub-node is defined as an action. d_{ip} ($1 \leq p \leq m_i$, m_i is the number of subnodes in judgment and processing nodes) is the time delay spent on the judgment or processing at node i, while d_{ip}^A, d_{ip}^B, ... are time delays spent on the node transition from node i to the next node. In this chapter, d_{ip}^A, d_{ip}^B, ... are set at zero time unit, d_{ip} of each judgment node is set at one time unit, d_{ip} of each processing node is set at five time units. We suppose that the trade in one day ends when GNP uses five or more time units, which means the trade in one day ends when GNP executes fewer than five judgment nodes and one processing node, or five

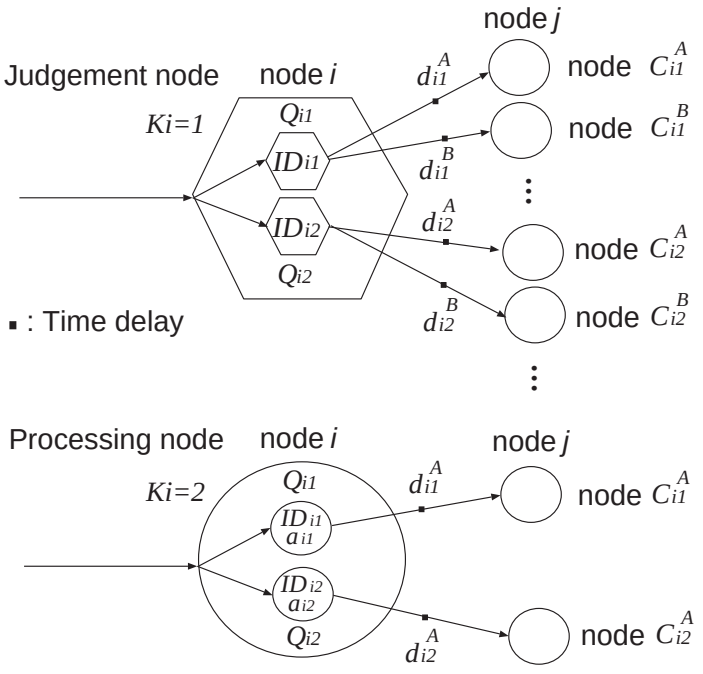


Figure 2.2: Node structure

judgment nodes. C_{ip}^{A}, C_{ip}^{B}, ... show the node number of the next node. Judgment node determines the upper suffix of the connection genes to refer to depending on the judgment result. If the judgment result is "B", GNP-Sarsa refers to C_{ip}^{B} and d_{ip}^{B}. Processing nodes always refer to C_{ip}^{A} and d_{ip}^{A} because processing nodes have no conditional branch.

2.3.2 Judgment and Processing Functions of GNP-Sarsa

The node transition of GNP-Sarsa starts from a start node and continues depending on the node connections and judgment results. Fig. 2.2 shows node structures of a judgment node and a processing node.

(1) Judgment node: When a current node i is a judgment node, first, one Q value is selected from Q_{i1}, ..., Q_{im_i} based on ϵ-greedy policy. That is, a maximum Q value among Q_{i1}, ..., Q_{im_i} is selected with the probability of 1-ϵ, or a random one is selected with the probability of ϵ. Then corresponding function (ID_{ip}) is selected. The gene ID_{ip} shows a technical index or a candlestick GNP judges at node i. Each technical index has its own IMX function shown in Fig. 2.3. x axis shows the value of each technical index, and the sections A, B, C, ... correspond to judgment results. Suppose Q_{i1} and the corresponding ID_{i1}=1 (judgment of rate of deviation) are selected, and if the rate

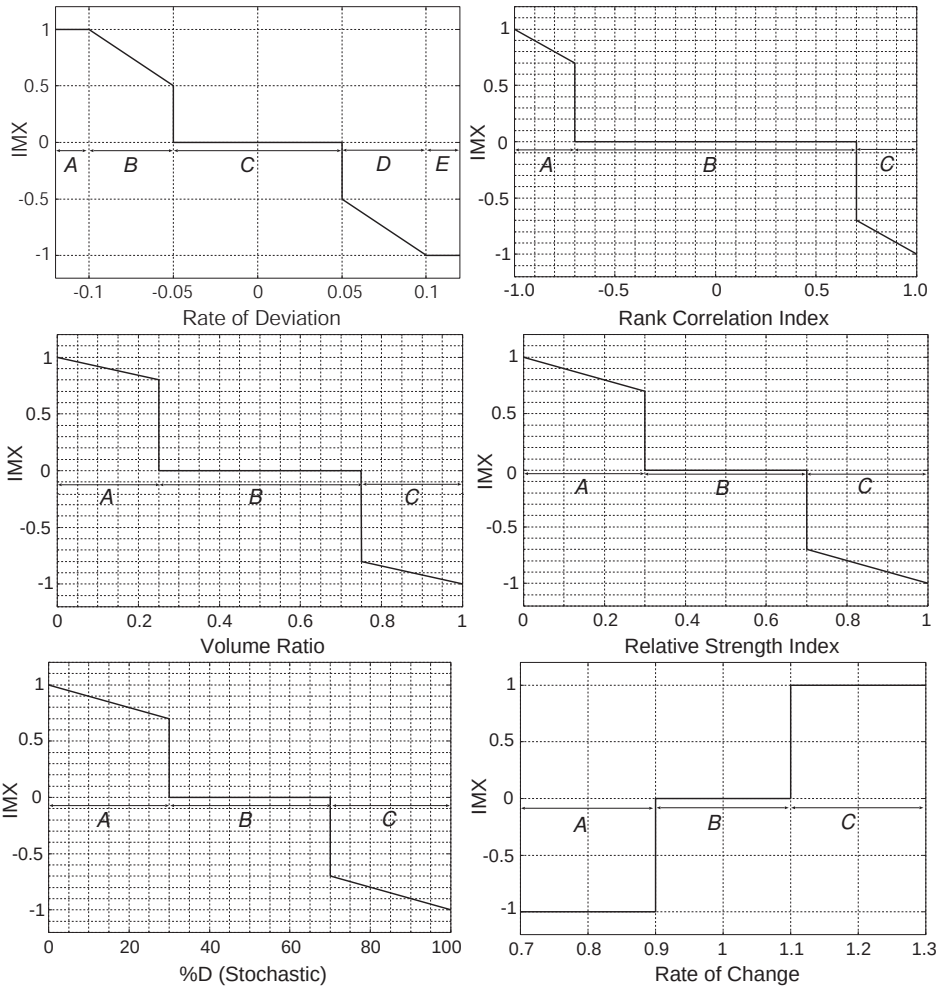


Figure 2.3: IMX functions in judgment nodes

is more than 0.1, the judgment result becomes E, and the next node number becomes C^E_{i1}. y axis shows the output of the IMX function and it is used at a processing node. However, the IMX output of golden cross, dead cross and MACD could be 1, 0 or -1 based on the cross of the lines, and the values correspond to judgment results A, B and C, respectively. Concretely speaking, for three days after a golden cross appears, the IMX output becomes 1, and for three days after a dead cross appears, it becomes -1, otherwise 0. Furthermore, for three days after MACD passes through the signal from the lower side to the upper side, the IMX output becomes 1, and for three days after it does from the upper to the lower, the IMX output becomes -1, otherwise it becomes 0. Generally, golden cross indicates buying signals and dead cross indicates selling signals, therefore, buying signals become stronger as the IMX output is close to 1, and

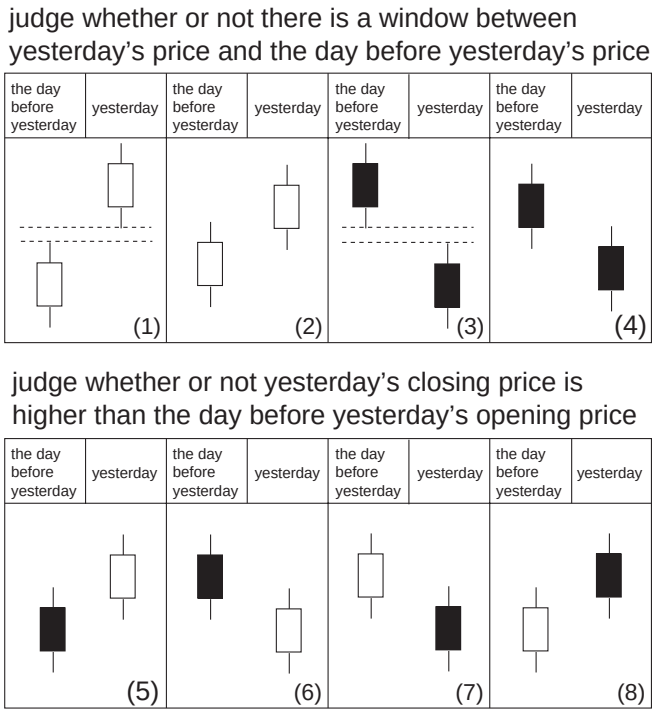


Figure 2.4: Candlestick chart patterns

selling signals become stronger as it is close to -1.

In this chapter, candlestick chart is used as one of judgment functions. As we know, candlestick chart has been winning international recognition for its good indication of stock prices, and it has been widely used as the means of indicating the fluctuations of the stocks. The proposed method has judgment nodes which check candlestick chart patterns. The judgment function of candlestick chart is executed as follows. When the selected sub-node has a judgment function of candlestick chart, GNP judges yesterday's candlestick and the candlestick of the day before yesterday. There are eight patterns of candlestick charts as shown in Fig. 2.4 according to two kinds of rules: (A) Judge whether there is a gap or not between yesterday's lowest price and the highest price of the day before yesterday, or between yesterday's highest price and the lowest price of the day before yesterday. (B) Judge whether or not yesterday's closing price is higher than the opening price of the day before yesterday. Especially, when the opening price equals to the closing price, the case is treated as black body candlestick. As an example, when the candlestick pattern is "3", GNP-Sarsa selects third branch to transfer to the next node. However, judgment nodes of candlestick chart do not have IMX

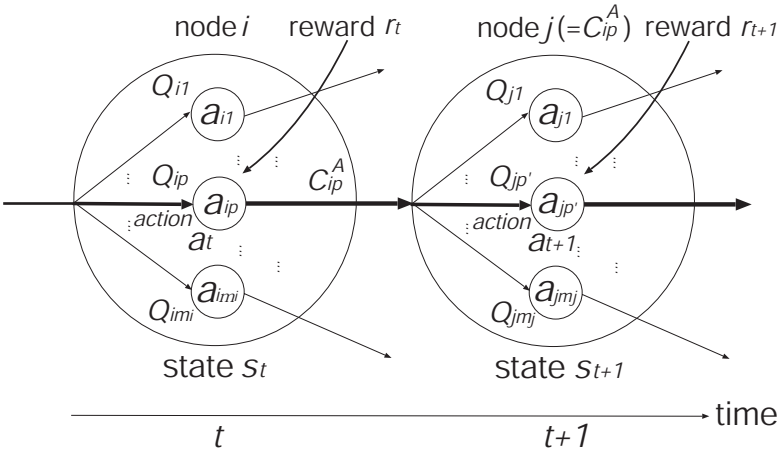


Figure 2.5: An example of node transition

function.

(2) Processing node: When a current node is a processing node, Q_{ip}, the corresponding ID_{ip} and a_{ip} are selected based on ϵ-greedy policy. The selected a_{ip} is a threshold for determining buying or selling stocks. We explain the procedure of buying and selling stocks using Fig. 2.5, where the current node at time t is a processing node.

1. First, one Q value is selected from Q_{i1}, ..., Q_{im_i} based on ϵ-greedy policy. That is, a maximum Q value among Q_{i1}, ..., Q_{imi} is selected with the probability of 1-ϵ, or a random one is selected with the probability of ϵ. Then, the corresponding a_{ip} is selected.

2. Calculate an average of the IMXs obtained at the judgment nodes executed in the node transition from the previous processing node to the current processing node.

$$A_t = \frac{1}{|I'|} \sum_{i' \in I'} IMX(i')$$

where, I' shows a set of suffixes of the judgment node numbers executed in the node transition from the previous processing node to the current processing node. IMX(i') shows an IMX output at node $i' \in I'$. However, when a judgment node of the candlestick chart was executed or an IMX output is zero at a judgment node of golden cross, deadcross and MACD, the node number is excluded from I' for calculating A_t.

3. Determine buying or selling:

   - In the case of ID_{ip}=0 (buy): if $A_t \geq a_{ip}$ and we do not have any stocks, GNP buys as much stocks as possible. Otherwise, GNP takes no action.
   - In the case of ID_{ip}=1 (sell): if $A_t < a_{ip}$ and we have stocks, GNP sells all the stocks. Otherwise, GNP takes no action.

4. The current node is transferred to the next node. If a_{ip} is selected, the next node number becomes C_{ip}^{A} .

The above procedure puts the information of the technical indices together into A_t, and GNP-Sarsa determines buying or selling stocks by comparing A_t with a_{ip}. Therefore, the points of this chapter are 1) to find appropriate a_{ip} in the processing nodes by evolution and Sarsa, and 2) to determine I' by evolution, in other words, what kinds of judgments (technical indices and candlestick charts) should be considered is determined automatically.

2.3.3 Learning Phase

First, we explain Sarsa algorithm briefly. Sarsa can obtain Q values which estimate the sum of the discounted rewards obtained in the future. Suppose an agent selects an action a_t at state s_t at time t, a reward r_t is obtained and an action a_{t+1} is taken at the next state s_{t+1}. Then $Q(s_t, a_t)$ is updated as follows.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

α is a step size parameter, and γ is a discount rate which determines the present value of future rewards: a reward received k time steps later is worth only γ^{k-1} times of the reward supposed to receive at the current step.

As described before, a state means the current node and an action means the selection of a sub-node. Here, we explain the procedure for updating Q value in this chapter.

1) At time t, GNP refers to Q_{i1}, ..., Q_{im_i} and selects one of them based on ϵ-greedy policy. Suppose that GNP selects Q_{ip} and the corresponding function ID_{ip}.

2) GNP executes the function ID_{ip}, gets the reward r_t and suppose the next node j becomes C_{ip}^{A}.

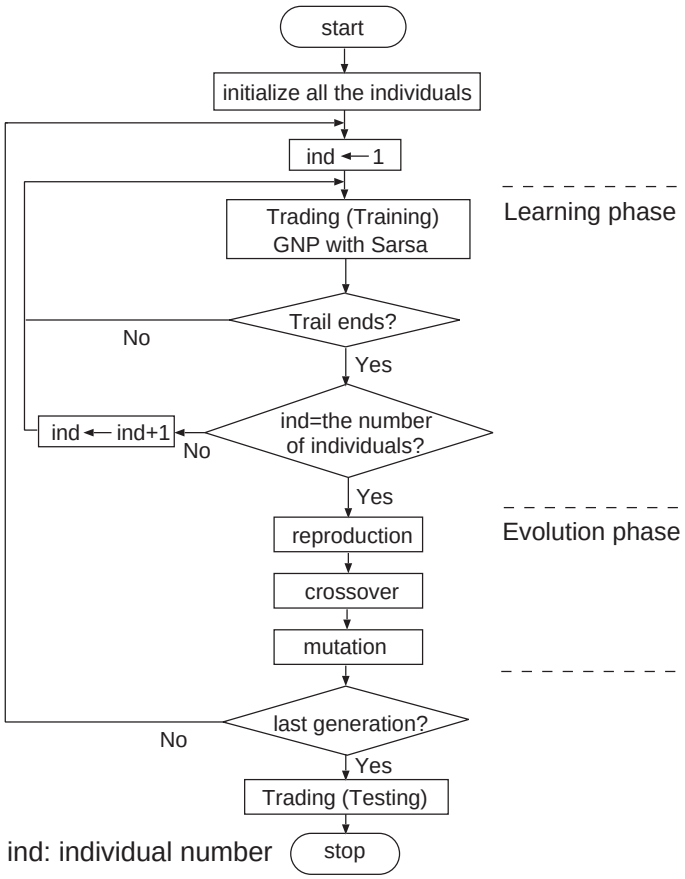


Figure 2.6: Flowchart of GNP-Sarsa

3) At time t+1, GNP selects one Q value in the same way as step1. Suppose that $Q_{jp'}$ is selected.

4) Q value is updated as follows.

$$Q_{ip} \leftarrow Q_{ip} + \alpha[r_t + \gamma Q_{jp'} - Q_{ip}]$$

5) $t \leftarrow t$+1, $i \leftarrow j$, $p \leftarrow p'$ then return step 2.

2.3.4 Evolution Phase

Fig. 2.6 shows the whole flowchart of GNP-Sarsa. In this sub-section, the genetic operators in the evolution phase are introduced. The role of evolution is to change graph structures and randomly change node parameters a_{ip}.

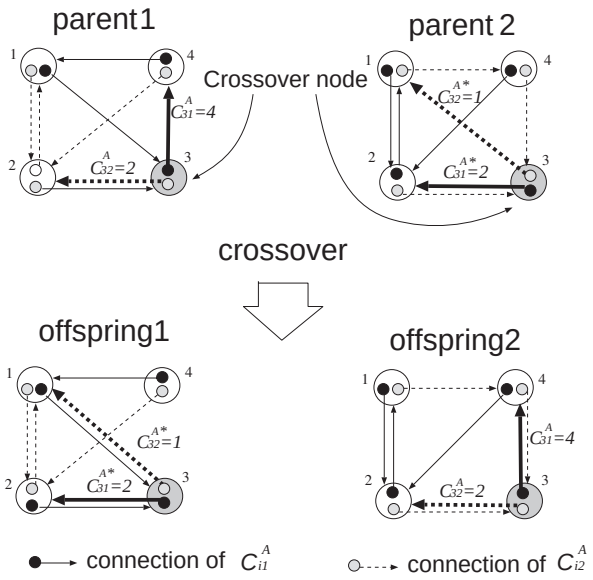


Figure 2.7: Crossover

Crossover

Crossover is executed between two parents and generates two offspring [Fig. 2.7]. The procedure of crossover is as follows.

1. Select two individuals using tournament selection twice and reproduce them as parents.
2. Each node is selected as a crossover node with the probability of P_c.
3. Two parents exchange the genes of the corresponding crossover nodes, i.e., the nodes with the same node number.
4. Generated new individuals become the new ones of the next generation.

Fig. 2.7 shows a crossover example of the graph structure with three processing nodes for simplicity. If GNP exchanges the genes of judgment nodes, it must exchange all the genes with suffix $A, B, C, \ldots$ simultaneously.

Mutation

Mutation is executed in one individual and a new one is generated [Fig. 2.8]. The procedure of mutation is as follows.

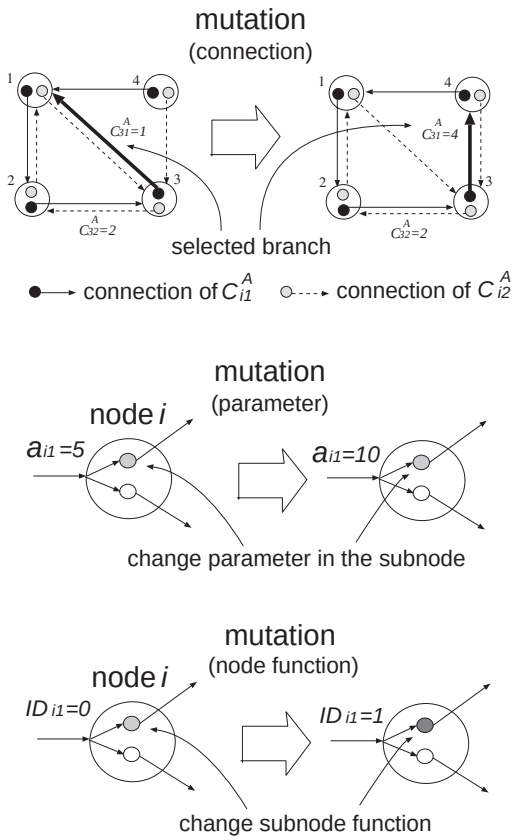


Figure 2.8: Mutation

1. Select one individual using tournament selection and reproduce it as a parent.

2. Mutation operation

   (a) change connection: Each node branch (C^A_{ip}, C^B_{ip}, ...) is selected with the probability of P_m, and the selected branch is reconnected to another node.

   (b) change parameters (a_{ip}): each a_{ip} is changed to other value with the probability of P_m.

   (c) change node function: Each node function (ID_{ip}) is selected with the probability of P_m, and the selected function is changed to another one.

3. Generated new individual becomes the new one of the next generation.

Table 2.1: Calculation periods of the technical index [day]

Technical index	period1	period2	period3
Rate of deviation	5	13	26
RSI	5	13	26
ROC	5	13	26
Volume ratio	5	13	26
RCI	9	18	27
Stochastics	12	20	30
Golden/Dead cross	5(short term)	26(long term)	
MACD	5(short term)	26(long term)	9(signal)

2.4 Simulation

To confirm the effectiveness of GNP-Sarsa, we carried out the trading simulations using 16 brands selected from the companies listed in the first section of Tokyo stock market in Japan (see Table 2.3). The simulation period is divided into two periods; one is used for training and the other is used for testing simulation.
Training: January 4, 2001—December 30, 2003 (737 days)
Testing: January 5, 2004—December 30, 2004 (246 days)

We suppose that the initial funds is 5,000,000 Japanese yen in both periods, and the order of buying or selling is executed at the opening of the trading day, i.e., we can buy and sell stocks with the opening price.

2.4.1 Fitness and Reward

Reward shows a capital gain of one trade (one set of buying and selling) and is used for learning. Fitness is the sum of the rewards obtained in the trading period.

Reward=selling price-purchase price

Fitness=Σ Reward

2.4.2 Conditions of GNP-Sarsa

GNP-Sarsa uses judgment nodes which judge the technical indices shown in Table 2.1 and candlestick charts. The technical indices are calculated using three kinds of calculation periods except Golden/Dead cross and MACD. Therefore, the number of kinds

Table 2.2: Simulation conditions

Number of individuals	300
Mutation	179
Crossover	120
Elite	1
Number of nodes	31
Judgement node	20
Processing node	10
Start node	1
Number of sub-node in each node	2
P_c	0.1
P_m	0.02
α	0.1
γ	0.4
ϵ	0.1

of judgment nodes is 21 (including one candlestick judgment). The number of processing functions is two: buying and selling. Table 2.2 shows simulation conditions. The total number of nodes in each individual is 31 including 20 judgment nodes, 10 processing nodes and one start node. However, the functions ID_{ip} in sub-nodes are determined randomly at the beginning of the first generation, and changed appropriately by evolution.

The initial connections between nodes are also determined randomly at the first generation. At the end of each generation, 179 new individuals are produced by mutation, 120 new individuals are produced by crossover, and the best individual is preserved. The other parameters are the ones showing good results in the simulations. The initial Q values are set at zero.

2.4.3 Simulation Results

First, 300 individuals are evolved for 300 generations using the training data. Fig. 2.9 shows the fitness curve of the best individual at each generation in the training term using the data of Toyota motor, and the line is the average over 30 independent simulations. From the Figure, we can see that GNP-Sarsa can obtain larger profits for the training data as the generation goes on. The fitness curves of the other companies have almost the same tendency as that of Toyota Motor.

Table 2.3: Profits in the test simulations

	Profit[yen](profit rate[%])		
Brand	GNP-Sarsa	GNP	Buy&Hold
Toyota Motor	522,333(10.4)	480,500(9.6)	520,000(10.4)
Mitsubishi Estate	444,733(8.9)	405,700(8.1)	664,000(13.3)
Showa Sell Sekiyu	263,100(5.3)	294,755(5.9)	319,200(6.4)
East Japan Railway	413,833(8.3)	491,500(9.8)	477,000(9.5)
NEC Corporation	36,600(0.7)	-126,150(-2.5)	-1,026,000(-20.5)
Fuji Heavy Ind.	217,133(4.3)	97,700(2.0)	-189,000(-3.8)
Sekisui House, Ltd.	582,466(11.6)	54,600(1.1)	264,000(5.3)
Mitsu & Co.	473,033(9.5)	118,450(2.4)	240,000(4.8)
Sony	148,733(3.0)	280,500(5.6)	150,000(3.0)
Tokyo Gas	669,733(13.4)	382,000(7.6)	372,000(7.4)
KDDI	199,400(4.0)	-76,600(-1.5)	-576,000(-11.5)
Tokyo Electric Power	570,266(11.4)	210,000(4.2)	262,500(5.3)
Daiwa House	612,633(12.3)	235,400(4.7)	32,000(0.6)
Nomura Holdings	366,033(7.3)	-293,785(5.9)	-985,500(-19.7)
Shin-Etsu Chemical	562,700(11.3)	7,250(0.1)	-264,000(-5.3)
Nippon Steel	469,866(9.4)	-27,350(0.5)	399,000(8.0)
Average	409,537(8.2)	158,404(3.2)	41,200(0.8)

Next, the test simulation is carried out using the best individual at the last generation in the training term. Table 2.3 shows the profits and losses in the testing term. The values in Table 2.3 are the average of the 30 independent simulations with different random seeds. For the comparison, the table also shows the results of Buy&Hold which is often considered to be a benchmark in trading stocks simulations. Buy&Hold buys as much stocks as possible at the opening of the market on the first day in the simulations, and sells all the stocks at the opening on the last day. From the table, the proposed method can obtain larger profits than Buy&Hold in the trade of 12 brands out of 16. By comparing with original GNP, the proposed method can get larger profits than traditional GNP in the trade of 13 brands out of 16. Especially, the stock prices of NEC, Fuji Heavy Ind., KDDI, Nomura Holdings, Shin-Etsu Chemical Co., Ltd. are down trend, so Buy&Hold always makes a loss, however the proposed method can obtain profits in five all brands.

Fig. 2.10 shows the change of the price of Toyota motor in the testing term and also shows typical buying and selling points by the proposed method. Fig. 2.11 shows the change of the funds as a result of the trading. From these figures, we can see that

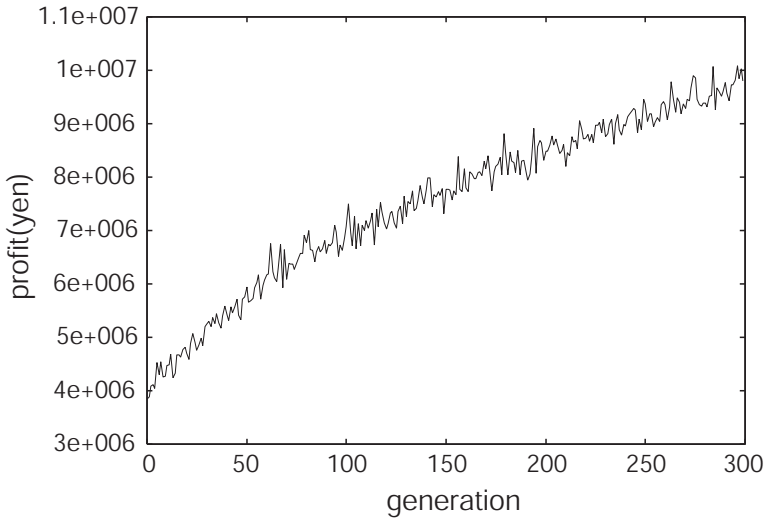


Figure 2.9: Fitness curve in the training period (Toyota Motor)

GNP-Sarsa can buy stocks at the lower points and sell at the higher points.

Fig. 2.12 shows the average ratio of the nodes used in the test period over 30 independent simulations in order to see which nodes are used and which are most efficient for stock trading model. The total number of node function is 23, while each processing node has a node number (0-1), and each judgment node has a node number (2-22). The x-axis shows the kinds of the nodes while the y-axis shows the average ratio of the used nodes. From the figure, we can see that the processing nodes are used to determine buying and selling stocks, and the judgment nodes of "Rate of deviation1" corresponding to period1 and "Volume ratio3" corresponding to period3 are frequently used.

Thus it can be said that GNP-Sarsa judges that these nodes are important to determine stock trading. GNP-Sarsa can automatically determine which nodes should be used in the current situation by evolving node functions and connections between nodes, in other words, GNP-Sarsa can optimize the combination of technical indices and candlestick charts used for stock trading model.

2.5 Summary

In this chapter, a stock trading model using GNP-Sarsa with important index and candlestick charts is proposed. First, a newly defined IMX function is assigned to each

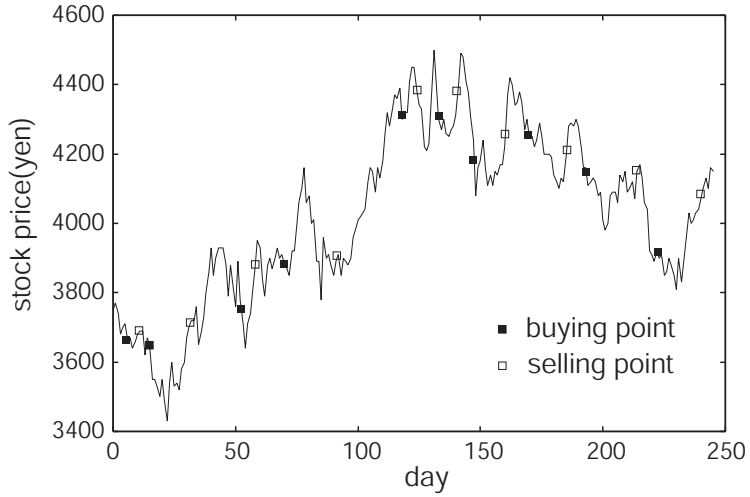


Figure 2.10: Stock price of Toyota Motor and typical buying and selling point in 2004 (test period)

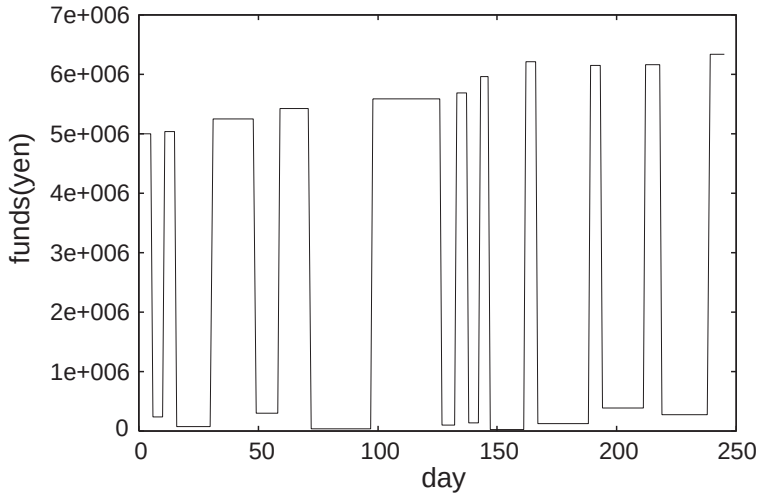


Figure 2.11: Change of funds in the test simulation (Toyota Motor)

technical index to tell GNP-Sarsa whether buying or selling stocks is recommended or not. Second, Sarsa learns Q values to select appropriate sub-nodes/functions used to judge the current stock price information and determine buying and selling timing. We carried out simulations using stock price data of 16 brands for four years. From the simulation results, it is clarified that the fitness becomes larger as the generation goes on and the profits obtained in the testing term are better than Buy&Hold in the simulations of 12 brands out of 16. By comparing with original GNP, the proposed method

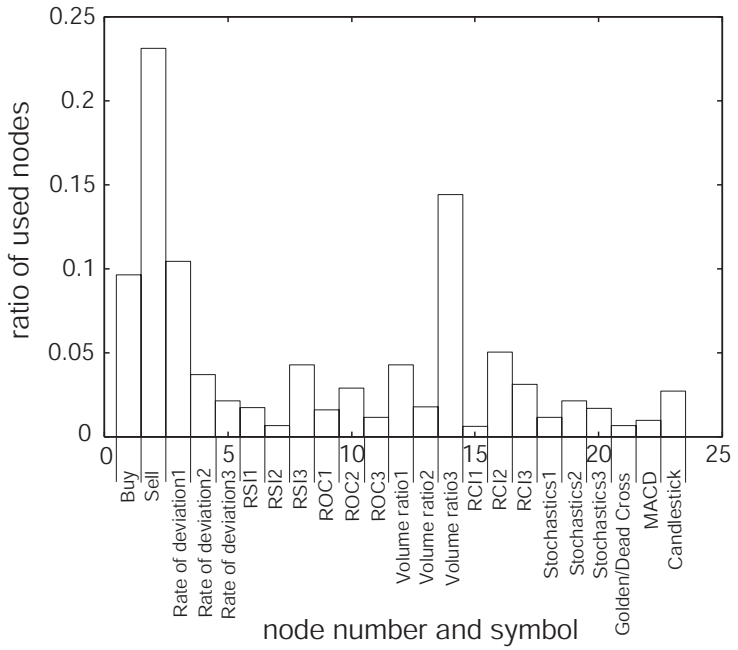


Figure 2.12: Ratio of nodes used by GNP-Sarsa in the test period (Toyota Motor)

can get larger profits than traditional GNP in the trade of 13 brands out of 16. When there is downtrend, Buy&Hold makes a loss in five brands, but the proposed method can obtain profits in five all brands.

There remain some problems to be solved. First, in this study, the calculation period of each technical index is fixed in advance. However, to improve the performance of the proposed method, we should develop a new method that can learn appropriate calculation periods. Next, it is necessary to consider the way of classifying the candlestick chart body type, and create more efficient judgment functions to judge the current stock price appropriately. Third, we should improve the performance of GNP-Sarsa model using time windows for adapting to the changing environment. Also, we will evaluate the proposed method comparing with other methods using many data of other brands.

Chapter 3

Real Time Updating Genetic Network Programming for Adapting to the Change of Stock Prices

3.1 Introduction

There have been increased a number of applications of Artificial Intelligence (AI) techniques, mainly Artificial Neural Networks (ANNs), Genetic Algorithm (GA) and Genetic Programming (GP), which have been applied to technical financial forecasting [46], [59] as they have the ability to deal with complex nonlinear problems and have the self-adaptation for different statistical distributions. Although these AI approaches possess the properties required for the technical financial forecasting, they have some inherent bottlenecks. For example, GP sometimes causes the bloating problem due to its tree structure. ANNs cannot be used to explain the causal relationships between input and output variables because of their black box nature.

In our former research, GNP was successfully applied to stock trading model by the combination with Sarsa learning (GNP-Sarsa) [60], and its applicability and efficiency has been confirmed. Based on the GNP-Sarsa algorithm, in this chapter, we propose a new method called Real Time Updating Genetic Network Programming (RTU-GNP) for adapting to the change of stock prices, especially. Generally speaking, there are two major approaches for predicting stock prices and determining the timing of buying or selling stocks: fundamental and technical analysis. The researches on stock price prediction and trading model using soft computing methods belong to technical analysis, which determines the timing of buying and selling stocks based on the technical

indices. The proposed method RTU-GNP also belongs to technical analysis.

In this chapter, we extend our previous work on GNP-Sarsa [60] and propose an algorithm that integrates the GNP-Sarsa and real time updating model in order to create an efficient stock trading system adapting to the change of the stock trend. There are three important points in this chapter:

- First, the RTU-GNP method makes a stock trading decision considering both the recommendable information of technical indices and the candlestick charts according to the real time stock prices.
- Second, a real time updating system has been firstly introduced to the GNP trading model considering the change of stock prices, where sliding windows are used in the system.
- The third point is that, to improve the performances of the previous GNP-Sarsa stock trading model, we propose a new method that can learn the appropriate function describing the relation between each technical index and the IMX. This is an important point that devotes to the enhancement of the GNP-Sarsa algorithm.

In addition, paper [40] only used GNP with candlestick chart, while paper [61] only used GNP with Importance Index and Actor-Critic as a basic algorithm. In this proposed method, the stock trading rules are created based on RTU-GNP using both candlestick chart and Importance Index as judgment functions to get more effective stock price information from the real stock market. Moreover, in the learning phase, we use Sarsa learning method for GNP individuals, which is different from the Actor-Critic learning method used in paper [61]. To confirm the effectiveness of the proposed trading system, we added more simulations and compared the experimental results with these two traditional methods under the same conditions. The results show that we can get more efficient trading rules and obtain much more profits by the proposed RTU-GNP method.

Section 3.2 describes the proposed RTU-GNP approach. Section 3.3 presents experimental environments, conditions and results using RTU-GNP method based on the real time model updating. The trading profits are presented and compared with the stock trading with traditional GNP methods and Buy&Hold method. Finally, Section 3.4 concludes this paper.

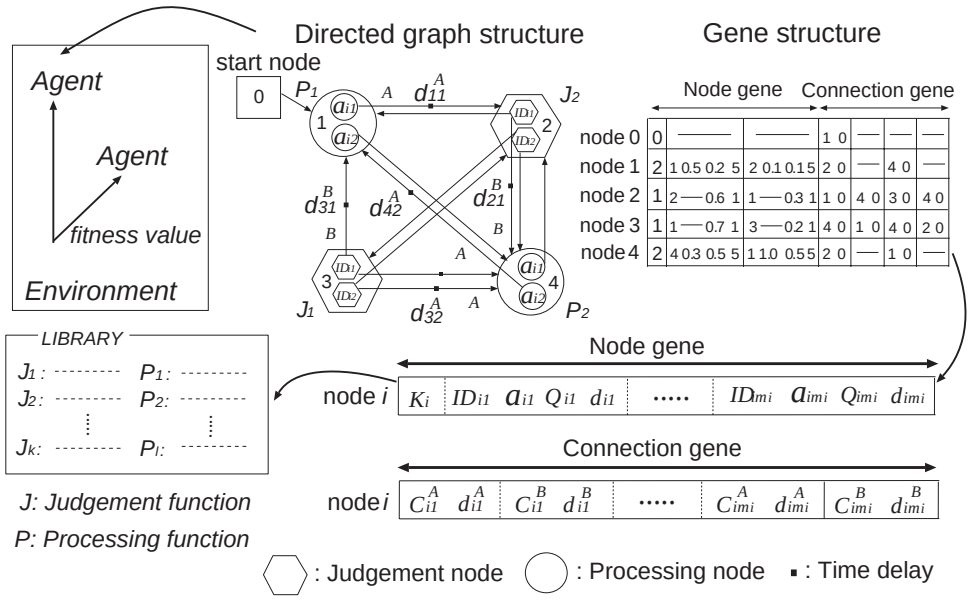


Figure 3.1: Basic structure of RTU-GNP

3.2 RTU-GNP Algorithm and Its Trading System

3.2.1 An Outline of RTU-GNP

Same as traditional GNP method, RTU-GNP is composed of a start node, judgment nodes and processing nodes, which are connected to each other. Fig. 3.1 shows a basic structure of RTU-GNP. The graph structure of RTU-GNP has some inherent characteristics such as compact structures and an implicit memory function that contribute to creating effective action rules.

Concretely speaking, K_i represents the node type, and ID_i represents an identification number of the node function. a_{ip} is a parameter which represents the threshold for determining buying or selling stocks in a processing node. Q_{ip} means Q value which is assigned to each state and action pair. In this method, "state" means the current node, and "action" means the selection of a sub-node. d_{ip} ($1 \leq p \leq m_i$, m_i is the number of subnodes in judgment and processing nodes) is the time delay spent on the judgment or processing at node i, while d_{ip}^A, d_{ip}^B, ... are time delays spent on the node transition from node i to the next node. In this paper, d_{ip}^A, d_{ip}^B, ... are set at zero time unit, d_{ip} of each judgment node is set at one time unit, d_{ip} of each processing node is set at five time units. C_{ip}^A, C_{ip}^B, ... show the node number of the next node. Judgment node determines

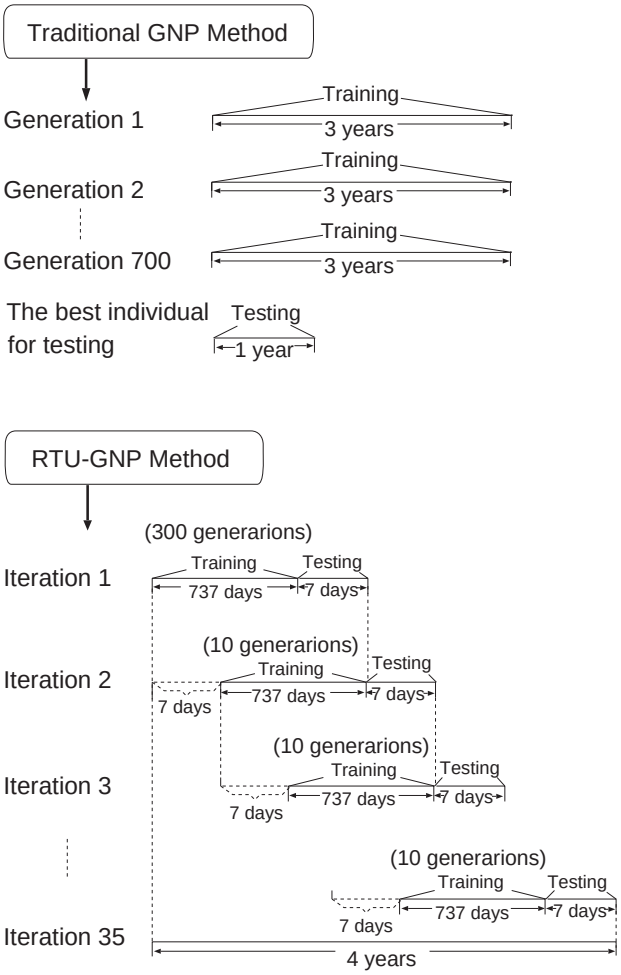


Figure 3.2: Comparison between RTU-GNP and traditional GNP

the upper suffix of the connection genes to refer to depending on the judgment result.

Like other evolutionary algorithms, crossover, mutation and selection are used as the genetic operators of RTU-GNP. The outline of evolution is described as follows:

(1) Initialize the first population and calculate the fitness of this population;

(2) Generate new individuals for the next generation by tournament selection and genetic operations;

(3) Calculate the fitness of the new individuals;

(4) Repeat 2-3 until the terminal condition meets.

3.2.2 Comparison Between RTU-GNP and Traditional GNP

As a key point of this paper, we propose RTU-GNP algorithm by using a newly defined method shown in Fig. 3.2 with real time model updating, which contributes to an improvement of the traditional GNP methods. Concretely speaking,

(1) Traditional Method

In our former research, we carried out the simulations using the real data of the stock market from January 4, 2001 to December 30, 2004 (4 years), i.e., we used the data from January 4, 2001 to December 30, 2003 (3 years, 737 days) for training, and the data from January 5, 2004 to December 30, 2004 (1 year, 246 days) for testing. As shown in Fig.2, we used the same data in 3 years for training using 700 generations, then we selected the best individual for 1 year testing.

(2) New Method with Real Time Model Updating

The data in stock markets are highly time-varying and changing every minute. Also, since the stock market data are given in an event-driven way, they are highly influenced by the indeterminate dealing. Because of these dynamic features, we propose a new method, which considers the time-related fluctuation and trend of stock prices well. In order to compare with traditional GNP, we carried out the simulations by the RTU-GNP method using the same data as traditional methods. The training and testing periods are updated in every iteration, however, the testing period in total is the same as traditional GNP. Concretely speaking, as shown in Fig. 3.2, we carried out the simulations using the real time model updating training&testing method. That is,

1) At the first iteration, we select the first 737 days (3 years) from the beginning (January 4, 2001) for training, and the next 7 days for testing. We use 300 generations for the 737 days' training since it is the first iteration, and GNP should be trained more to perform well.

2) At the second iteration, we shift 7 days from the beginning and select another 737 days for training. Like the first iteration, the next 7 days are selected for testing. Thus, there is 7 days difference between iteration 1 and iteration 2. After the second iteration, the number of generations for training is just small 10, because the training could use the evolved GNPs in the previous generation as initial GNPs.

3) Repeat such training&testing simulations by shifting 7 days every iteration until the terminal day reaches (December 30, 2004).

4) As a result, RTU-GNP could be used for the online trading, where GNPs are evolved every time the stock prices change.

By this setting of simulation periods, we get the same testing period (245 days in total) as tradition GNP by shifting 7 days for 35 times. In addition, we also changed the 7 days shifting (one week) to 30 days (one month) for another comparison to get an appropriate testing period. The simulation results are shown in Fig. 3.12, Fig. 3.13 and Table 3.3.

3.2.3 Judgment and Processing Functions of RTU-GNP

The node transition of RTU-GNP starts from a start node and continues depending on the node connections and judgment results.

(1) An introduction of technical indices and candlestick chart: Each judgment node uses one of the following technical indices for its judgment: Rate of Deviation from moving average (ROD), Relative Strength Index (RSI), Rate of Change (ROC), Volume Ratio, Rank Correlation Index (RCI), Stochastics, Golden/Dead cross and Moving Average Convergence and Divergence (MACD).

A candlestick chart analysis was used in order to elicit technical knowledge in this study. The candlestick chart is a useful tool to visualize the stock prices so that investors can detect patterns which can be used to predict future stock price movements. As illustrated in Fig. 3.3, the candlestick chart consists of a rectangle and two shadow lines. The rectangle which is called the real body indicates the difference between the opening value and the closing value of the stock. If the real body of the candlestick chart shows that the opening value is higher than the closing value, the candlestick chart is called black candlestick. On the other hand, if the closing value is higher than the opening value, the candlestick chart is called white candlestick chart. The white candlestick implies the rising signal of the stock price and the black candlestick implies the falling signal. The stock price patterns which are represented by the candlestick chart shapes give important clues to predict future stock price movements. Thus, the technical knowledge from the candlestick chart was used as a key information for predicting stock price movements in this study. The knowledge from the candlestick chart analysis shows that if a certain pattern occurs, then the stock price will increase (or decrease).

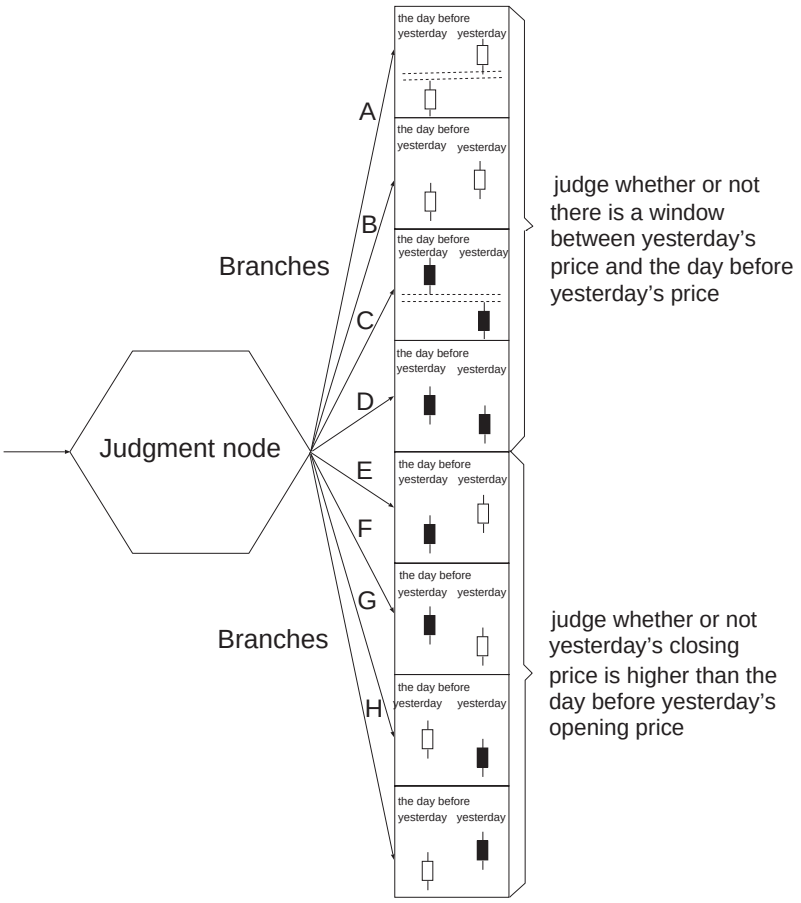


Figure 3.3: Candlestick chart patterns

(2) Judgment function: When the current node i is a judgment node, first, one Q value is selected from Q_{i1}, ..., Q_{im_i} based on ϵ-greedy policy. That is, a maximum Q value among Q_{i1}, ..., Q_{im_i} is selected with the probability of 1-ϵ, or a random one is selected with the probability of ϵ. Then corresponding function (ID_{ip}) is selected. The gene ID_{ip} shows a technical index or a candlestick GNP judges at node i. To improve the performance of RTU-GNP algorithm, we propose a method in this paper that can learn an appropriate function by describing the relation between the value of each technical index and the value of IMX. That means, judgment results are determined by the evolution of IMX functions. Concretely speaking, each technical index has its own IMX function as shown in Fig. 3.4 and Fig. 3.5, where x axis shows the value of each technical index, and the sections A, B, C ... correspond to judgment results. Suppose Q_{i1} and the corresponding ID_{i1}=1 (judgment of rate of deviation) are selected, and if the rate

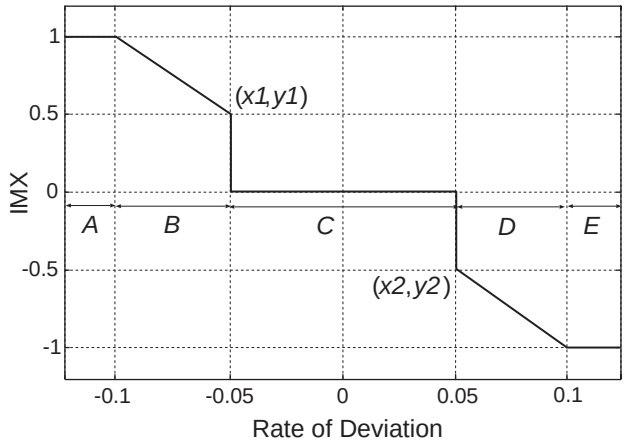


Figure 3.4: IMX functions in judgment nodes (In case of ROD)

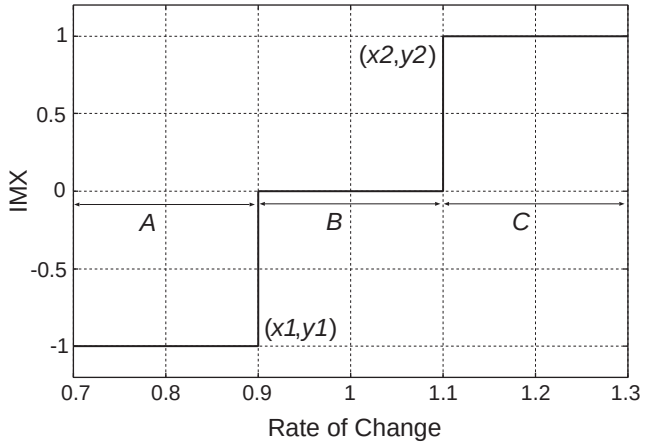


Figure 3.5: IMX functions in judgment nodes (In case of ROC)

is more than 0.1, the judgment result becomes E, and the next node number becomes C_{i1}^{E}. On the other hand, y axis shows the output of the IMX function and it is used at a processing node. From the figures, we can see that there are two points (left and right) which could be changed by evolution. For example, in the figure of ROC the left point is described by (x_1, y_1) and the right one is described by (x_2, y_2) respectively. Suppose Q_{i1} and the corresponding ID_{i1}=1 (judgment of ROC) are selected, and if the value of ROC is more than x_2, the judgment result becomes C, and the next node number becomes C_{i1}^{C}. In the evolution phase, each value of x_1, y_1, x_2 and y_2 of IMX functions is changed to other values with the probability of p_m. However, the IMX output of golden cross, dead cross and MACD could be 1, 0 or -1 based on the cross of the lines,

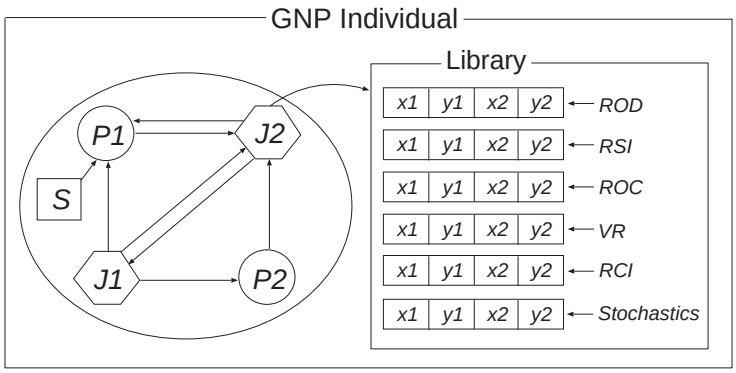


Figure 3.6: Parameters of IMX function of each technical index in the judgment node

and the values correspond to judgment results A, B and C, respectively. Concretely speaking, for three days after a golden cross appears, the IMX output becomes 1, and for three days after a dead cross appears, it becomes -1, otherwise 0. Generally, golden cross indicates buying signals and dead cross indicates selling signals, therefore, buying signals become stronger as the IMX output is close to 1, and selling signals become stronger as it is close to -1. As shown in Fig. 3.6, when the current judgment function is technical index, GNP refer to the parameters x_1, y_1, x_2 and y_2 of each technical index to determine the shape of IMX function, i.e., judgment results are determined by the IMX functions consequently.

In order to make an appropriate shape of IMX function, we referred to much stock trading knowledge and experience from the stock markets, especially for the analysis of technical indices. For example, as shown in Fig. 3.4, the shape of IMX function for ROD (Rate of Deviation) can be determined by the following procedure:

- Step 1: The points (-0.1, 1.0) and (0.1, -1.0) are fixed in advance by the knowledge of technical indices.
- Step 2: Set the following constraint conditions:
  (1) $-0.1 \leq x_1 \leq 0.0$, (2) $0.0 \leq y_1 \leq 1.0$, (3) $0.0 \leq x_2 \leq 0.1$, (4) $-1.0 \leq y_2 \leq 0.0$.
- Step 3: Set the initial value of x_1, y_1, x_2 and y_2, then the initial shape of IMX is determined.
- Step 4: Evolve the value of x_1, y_1, x_2 and y_2 of IMX functions by mutation with the probability of p_m under the above constraints.

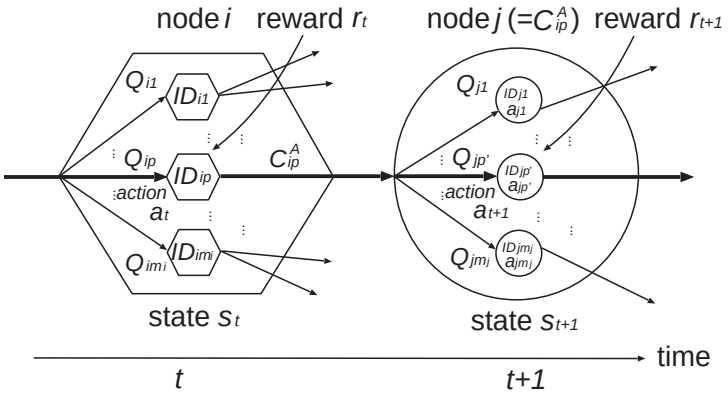


Figure 3.7: An example of node transition

Especially, even if the shape of IMX function of each technical index is determined by the same procedure, the constraint conditions of each index in step 2 is different, i.e., the evolving range for x_1, y_1, x_2 and y_2 are set by different constraint parameters for different technical indices.

As shown in Fig. 3.3, the candlestick chart is used as one of the judgment functions in this paper. The proposed method has judgment nodes which check candlestick chart patterns. The judgment function of the candlestick chart is executed as follows. When the selected sub-node has a judgment function of the candlestick chart, GNP judges yesterday's candlestick and the candlestick of the day before yesterday. There are eight candlestick chart patterns from "A" to "H" as shown in Fig. 3.3 according to two kinds of rules: (A) Judge whether there is a gap or not between yesterday's lowest price and the highest price of the day before yesterday, or between yesterday's highest price and the lowest price of the day before yesterday. (B) Judge whether or not yesterday's closing price is higher than the opening price of the day before yesterday. As an example, when the candlestick pattern is "C", RTU-GNP selects third branch to transfer to the next node.

(3) Processing function: When the current node is a processing node, Q_{ip} ($1 \leq p \leq m_i$, m_i is the number of subnodes in judgment and processing nodes), the corresponding ID_{ip} and a_{ip} are selected based on ϵ-greedy policy. The selected a_{ip} is the threshold for determining buying or selling stocks. We explain the procedure of buying and selling stocks using Fig. 3.7.

1) First, one Q value is selected from Q_{i1}, ..., Q_{im_i} based on ϵ-greedy policy. That

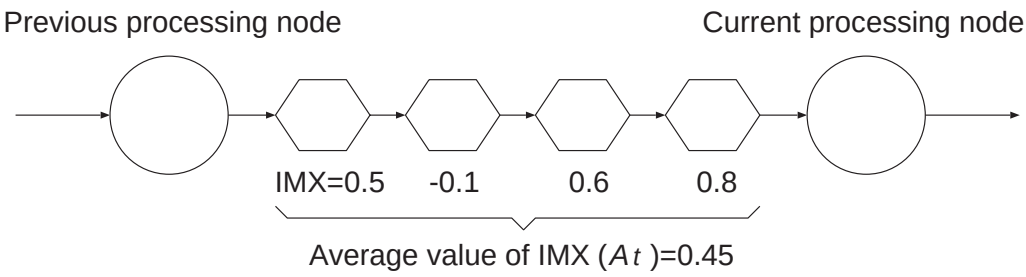


Figure 3.8: Calculation of the average value of IMX

is, a maximum Q value among Q_{i1}, ..., Q_{imi} is selected with the probability of 1-ϵ, or a random one is selected with the probability of ϵ. Then, the corresponding a_{ip} is selected.

2) Calculate an average of the IMXs obtained at the judgment nodes executed in the node transition from the previous processing node to the current processing node, as shown in Fig. 3.8.

$$A_t = \frac{1}{|I'|} \sum_{i' \in I'} IMX(i'),$$

where, I' shows a set of suffixes of the judgment nodes executed in the node transition from the previous processing node to the current processing node. IMX(i') shows an IMX output at node $i' \in I'$. However, when a judgment node of the candlestick chart was executed or an IMX output is zero at the judgment node of golden cross, deadcross and MACD, the node number is excluded from I' for calculating A_t.

3) Determine buying or selling:

- In the case of ID_{ip}=0 (buy): if $A_t \geq a_{ip}$ and we do not have any stocks, GNP buys as much stocks as possible. Otherwise, GNP takes no action.
- In the case of ID_{ip}=1 (sell): if $A_t < a_{ip}$ and we have stocks, GNP sells all the stocks. Otherwise, GNP takes no action.

4) The current node is transferred to the next node. If ID_{ip} is selected, the next node number becomes C^A_{ip} .

The above procedure puts the information of the technical indices together into A_t, and RTU-GNP determines buying or selling stocks by comparing A_t with the threshold a_{ip}. Therefore, the important points are 1) to find appropriate a_{ip} in the processing nodes by evolution and reinforcement learning, and 2) to determine I' by evolution, in other

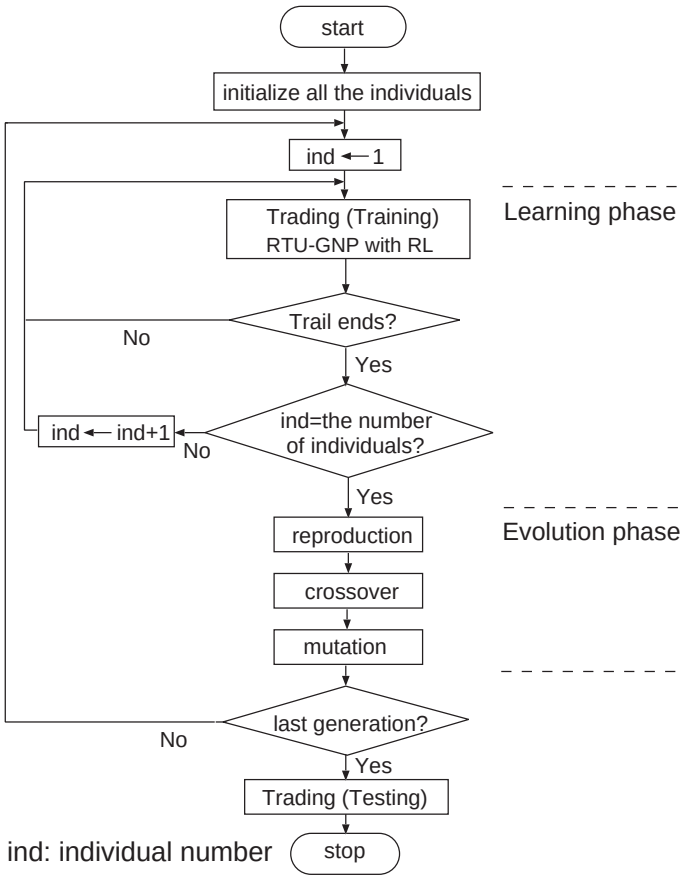


Figure 3.9: Flowchart of RTU-GNP

words, what kinds of judgments (technical indices and candlestick charts) should be considered is determined automatically.

3.2.4 Evolution of RTU-GNP

Fig. 3.9 shows the whole flowchart of RTU-GNP. In this sub-section, the genetic operators in the evolution phase are introduced. RTU-GNP with RL also has three kinds of genetic operators, i.e., selection, crossover and mutation. All of them except for mutation are the same as the original GNP. In RTU-GNP with RL, mutation operation could be executed not only on the connections between nodes but also on the node functions and parameters.

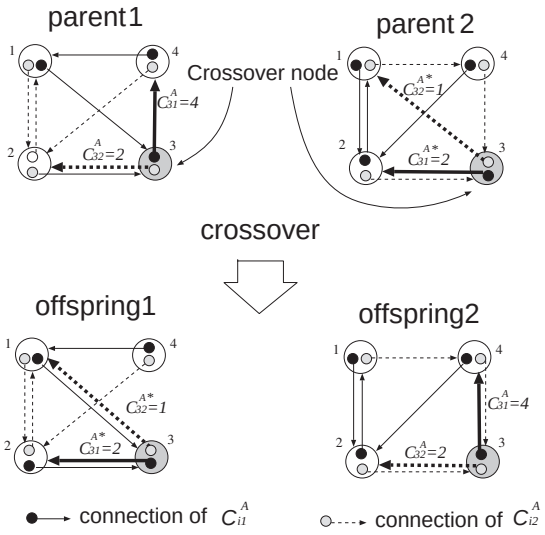


Figure 3.10: Crossover

Crossover

Crossover is executed between two parents and generates two offspring [see Fig. 3.10]. The procedure of crossover is as follows.

- Select two individuals using tournament selection twice and produce them as parents.
- Each node is selected as a crossover node with the probability of P_c.
- Two parents exchange the genes of the corresponding crossover nodes, i.e., the nodes with the same node number.
- Generated new individuals become the new ones of the next generation.

Fig. 3.10 shows a crossover example of the graph structure with four processing nodes for simplicity. If GNP exchanges the genes of judgment nodes, it must exchange all the genes with suffix A, B, C, ... simultaneously.

Mutation

Mutation is executed in one individual and a new one is generated [see Fig. 3.11]. The procedure of mutation is as follows.

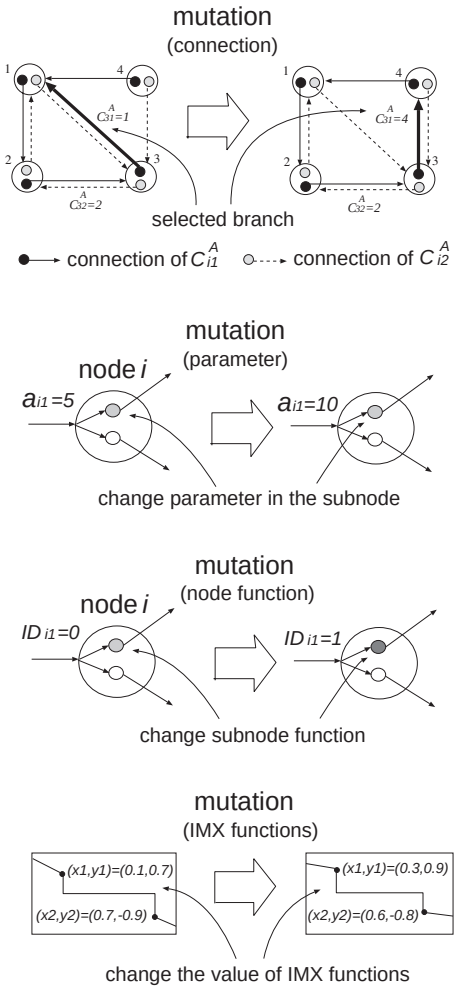


Figure 3.11: Mutation

- Select one individual using tournament selection as a parent.
- Mutation operation
  - change connection: Each node branch (C^A_{ip}, C^B_{ip}, ...) is selected with the probability of P_m, and the selected branch is reconnected to another node.
  - change parameters (a_{ip}): each a_{ip} is changed to the other value with the probability of P_m.
  - change node function: Each node function (ID_{ip}) is selected with the probability of P_m, and the selected function is changed to another one.
  - change x_1, y_1, x_2 and y_2 of IMX functions: each value is changed to other

value with the probability of P_m.

- Generated new individual becomes the new one of the next generation.

3.2.5 Learning of RTU-GNP

In this paper, we use RTU-GNP with Sarsa learning which is one kind of reinforcement learning methods. Sarsa can obtain Q values which estimate the sum of the discounted rewards obtained in the future. Suppose an agent selects an action a_t at state s_t at time t, a reward r_t is obtained and an action a_{t+1} is taken at the next state s_{t+1}. Then $Q(s_t, a_t)$ is updated as follows.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

α is a step size parameter, and γ is a discount rate. The state means the current node and action means the selection of the sub-node. The procedure for updating Q value in this chapter is same as GNP-Sarsa.

3.3 Simulation

To confirm the effectiveness of RTU-GNP, we carried out the trading simulations using 20 brands selected from the companies listed in the first section of Tokyo stock market in Japan (see Table 3.3). The simulation period is divided into two periods; one is used for training and the other is used for testing. We use the real data of the stock market from January 4, 2001 to December 30, 2004. As a key point of this paper, we have already described the new method shown in Fig. 3.2 with real time model updating in Section 3.2.2, which contributes to an improvement of the traditional simulation methods.

We can see that the advantage of the proposed method is to consider the changes of the trend of stock prices and update GNP model depending on their changes. Thus, RTU-GNP method works well, when applied to the stock market, especially in the financial stock market with its trend being changed. We suppose that the initial fund is 5,000,000 Japanese yen, and the order of buying or selling is executed at the opening of the trading day, i.e., we can buy and sell stocks with the opening price.

Table 3.1: Calculation periods of the technical index [day]

Technical index	Technical index used in GNP		
Rate of Deviation	5ROD	13ROD	26ROD
Relative Strength Index	5RSI	13RSI	26RSI
Rate of Change	5ROC	13ROC	26ROC
Volume Ratio	5VR	13VR	26VR
Rank Correlation Index	9RCI	18RCI	27RCI
Stochastics	12S	20S	30S
Golden/Dead Cross	5G/D	26G/D	—
Moving Average Convergence and Divergence	5MACD	9MACD	26MACD

Table 3.2: Simulation conditions

Number of individuals=300 (mutation=179, crossover=120, elite=1)
Number of nodes=31 (judgement node=20, processing node=10, start node=1) Number of sub-nodes in each node=2 P_c=0.1, P_m=0.02, α=0.1, γ=0.3, ϵ=0.1

3.3.1 Fitness and Reward

Reward in RL shows a capital gain of one trade (one set of buying and selling), while the fitness in evolution is the sum of the rewards obtained in the trading period.

Reward=selling price-purchase price

Fitness=ΣReward

3.3.2 Settings of RTU-GNP

RTU-GNP uses judgment nodes which judge the technical indices shown in Table 3.1 and candlestick chart. The technical indices are calculated using three kinds of calculation periods except Golden/Dead cross and MACD. Therefore, the number of kinds of judgment nodes is 21 (including one candlestick judgment). The number of processing functions is two: buying and selling. Table 3.2 shows simulation conditions. The total number of nodes in each individual is 31 including 10 processing nodes, 20 judgment nodes and one start node. However, the functions ID_{ip} in sub-nodes are determined randomly at the beginning of the first generation, and changed appropriately by evo-

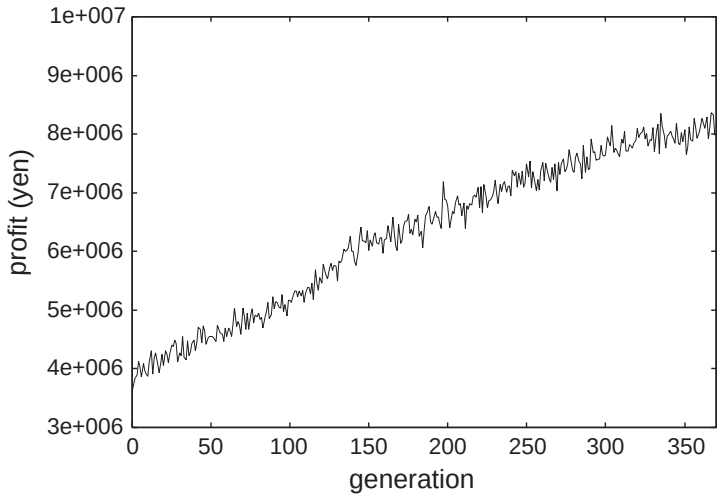


Figure 3.12: Fitness curve in the training period with 30-day shifting (Sony)

lution.

The initial connections between nodes are also determined randomly at the first generation. At the end of each generation, 179 new individuals are produced by mutation, 120 new individuals are produced by crossover, and the best individual is preserved. The number of sub-nodes in each node is two. The other parameters are the ones showing good results in the simulations. The initial Q values are set at zero.

3.3.3 Simulation Results

First, 300 individuals are evolved for 300 generations using the training data, which is shifted for 7 days in every iteration. From the second to the last iteration, individuals are evolved only for 10 generations, because the training data do not have so much difference from the previous iteration. Fig. 3.12 and Fig. 3.13 shows the fitness curve of the best individual at each generation in the training term using the data of Sony company in two kinds of testing time settings (7 days and 30 days respectively), and the line is the average over 30 independent simulations. From the figures, we can see that RTU-GNP with reinforcement learning can obtain larger profits for the training data as the generation goes on. The fitness curves of the other companies have almost the same tendency as that of Sony company. By comparing the fitness curves of Fig. 3.12 and Fig. 3.13, we can easily find that the obtained profit by 7 days testing is much higher than the profit obtained by 30 days testing, which means that 7 days testing can

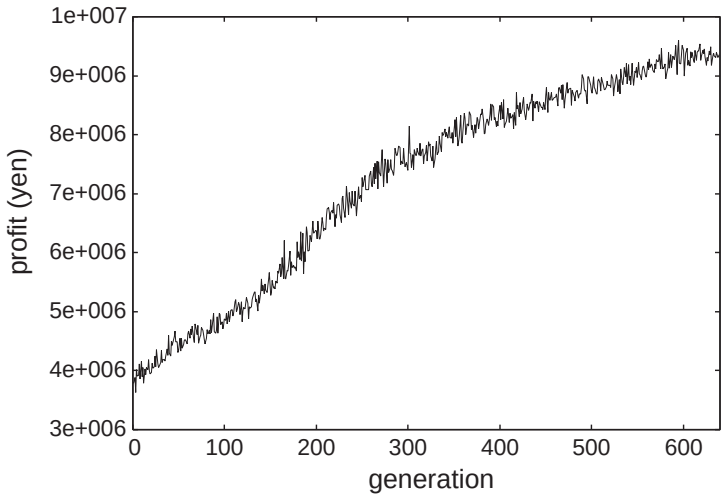


Figure 3.13: Fitness curve in the training period with 7-day shifting (Sony)

be more sensitive than 30 days adapting to the change of the trend of stock prices.

Next, the test simulation is carried out using the best individual at the last generation in the training term. In other words, the best individual obtained at each iteration is used for evaluating its 7 days' testing data. Table 3.3 shows the profits and losses in the testing term. The values in Table 3.3 are the average of the 30 independent simulations with different random seeds. For the comparison, the table also shows the results of GNP-Sarsa, GNP with Actor Critic, GNP with Candlestick which were proposed in the previous works. Also, we compare the results of the proposed method with Buy&Hold which is often considered to be a benchmark in trading stocks simulations. Buy&Hold buys as much stocks as possible at the opening of the market on the first day in the simulations, and sells all the stocks at the opening on the last day. From the table, the proposed method can obtain larger profits than Buy&Hold in the trade of 18 brands out of 20. By comparing with GNP-Actor Critic, the proposed method can get larger profits than GNP-Actor Critic in the trade of 15 brands out of 20. By comparing with GNP-Candlestick, the RTU-GNP method can get higher profit than GNP-Candlestick method in the trade of 19 brands out of 20. When comparing with GNP-Sarsa, which was introduced in Chapter 2, the RTU-GNP method can get higher profit than GNP-Sarsa in the trade of 15 brands out of 20. Especially, the stock prices of NEC, Fuji Heavy Ind., KDDI, Nomura Holdings, Shin-Etsu Chemical Co. and Tokyo Electric Power are down trend, so, Buy&Hold always makes a loss, however the proposed method can

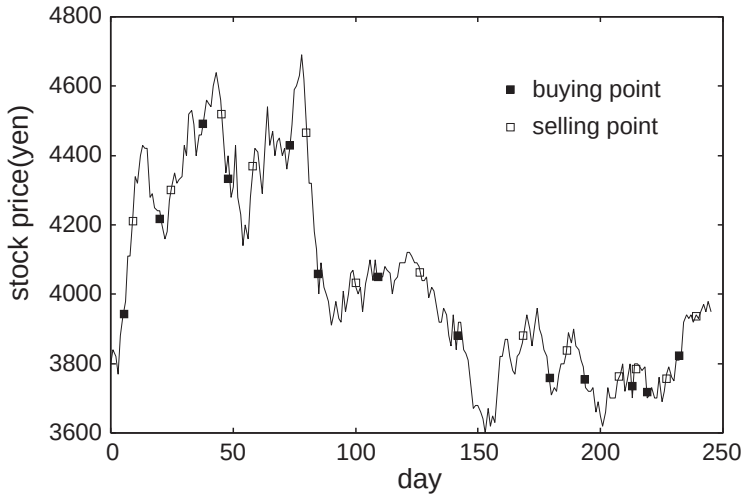


Figure 3.14: Stock price of Sony company and typical buying and selling point in 2004 (test period)

obtain profits in all six brands. From Table 3.3, we can also see that even if we use the same RTU-GNP method, the diffent time settings will affect the performance of GNP and the final profit, i.e., RTU-GNP with 7-day testing outperforms the 30-day testing, which means short-term testing is more sensitive than long-term testing in this case.

Fig. 3.14 shows the change of the price of Sony company in the testing term and also shows typical buying and selling points by the proposed method. From this figure, we can see that RTU-GNP can buy stocks at the lower points and sell at the higher points.

Fig. 3.15 shows the average ratio of the nodes used in the test period over 30 independent simulations in order to see which nodes are used and which are most efficient for stock trading models. The total number of node function is 23, where each processing node has a node number (1-2), and each judgment node has a node number (3-23). The x-axis shows the kinds of the nodes while the y-axis shows the average ratio of the used nodes. From the figure, we can see that the processing nodes are used to determine buying and selling stocks, and the judgment nodes of "ROC2" and "Stochastics2" are frequently used. Thus it can be said that RTU-GNP judges that these nodes are important to determine stock trading. RTU-GNP can automatically determine which nodes should be used in the current situation by evolving node functions and connections between nodes, in other words, RTU-GNP can optimize the combination of technical indices and candlestick charts used for the stock trading model.

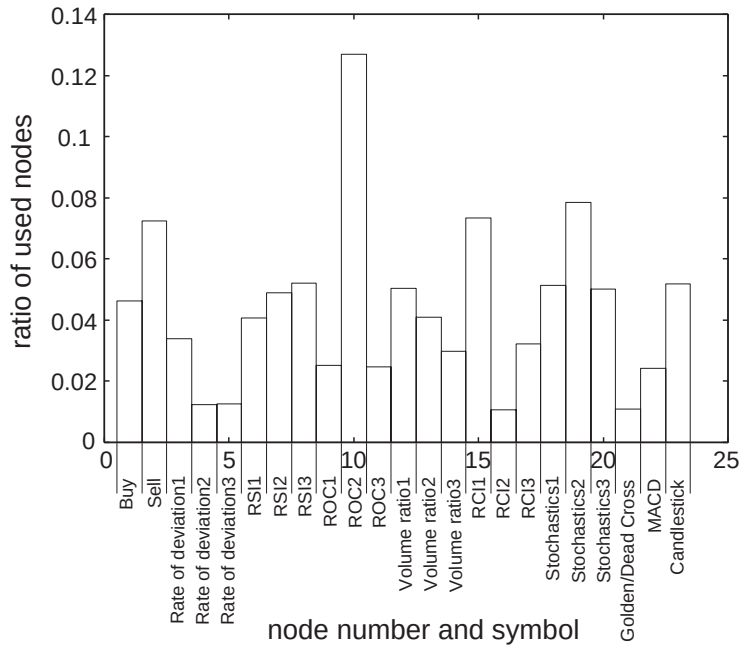


Figure 3.15: Ratio of nodes used by RTU-GNP in the test period (Sony)

3.3.4 Discussion

As an enhancement model of GNP-Sarsa, which was proposed in the previous chapter, RTU-GNP created a more effective model for adapting to the change of stock prices. Generally speaking, there are two important points that contributed to the improvements, and one of them is the real-time updating system, where sliding windows are used in both training and testing periods. Another point is the enhanced function of Importance Index, i.e., it can learn the appropriate function describing the relation between each technical index and the corresponding Importance Index by the evolution. Since the data in stock markets are highly time-variant and changing every minute, in financial practice, the proposed RTU-GNP model can learn the hidden interactions among variables and make more profits for the investors.

3.4 Summary

In this chapter, a stock trading model using RTU-GNP with important index and candlestick charts is proposed, where GNP-Sarsa can continue to work depending on the changes of the stock market, in other words, GNP-Sarsa can be adaptable to the fluc-

tuation of the stock market. First, a newly defined IMX function is assigned to each technical index to tell RTU-GNP whether buying or selling stocks is recommended or not. Second, RL learns Q values to select appropriate sub-nodes/functions, to judge the current stock price information and to determine buying and selling timing. We carried out the newly proposed method using stock price data of 20 brands for four years. From the simulation results, it is clarified that the fitness becomes larger as the generation goes on and the profits obtained in the testing term are better than Buy&Hold in the simulations of 18 brands out of 20. By comparing with GNP-Actor Critic, the proposed method can get larger profits than GNP-Actor Critic in the trade of 15 brands out of 20. By comparing with GNP-Candlestick, the RTU-GNP method can get higher profit than GNP-Candlestick method in the trade of 19 brands out of 20. When comparing with GNP-Sarsa, which was introduced in Chapter 2, the RTU-GNP method can get higher profit than GNP-Sarsa in the trade of 15 brands out of 20. Especially, when there is downtrend, Buy&Hold makes a loss in six brands, but the proposed method can obtain profits in six all brands.

There remain some problems deserve further investigation. First, in this chapter, the calculation period of each technical index is fixed in advance. However, to improve the performance of the proposed method, we should develop a new method that can learn appropriate calculation periods. Next, it is necessary to create more efficient judgment functions to judge the current stock price appropriately. Finally, we should develop extended methods to deal with multi brands in a portfolio simultaneously.

Table 3.3: Profits in the test simulations (Profit[yen](profit rate[%]))

Brand	RTU-GNP(7)	RTU-GNP(30)	GNP-Sarsa	GNP-Actor Critic	GNP-Candlestick	Buy&Hold
Toyota Motor	496,566(9.9)	502,733(10.1)	444,466(8.9)	495,350(9.9)	517,300(10.3)	520,000(10.4)
Mitsubishi Estate	630,433(12.6)	284,733(5.7)	261,033(5.2)	551,050(11.0)	615,300(12.3)	664,000(13.3)
Showa Sell Sekiyu K.K.	397,900(8.0)	375,233(7.5)	301,833(6.0)	295,000(5.9)	192,533(3.9)	319,200(6.4)
NEC Corporation	94,166(1.9)	43,400(0.9)	41,233(0.8)	-184,000(-3.7)	-69,500(-1.4)	-1,026,000(-20.5)
Fuji Heavy Ind.	165,500(3.3)	138,800(2.8)	76,066(1.5)	208,650(4.2)	-7,833(-0.2)	-189,000(-3.8)
Mitsu & Co.	486,900(9.7)	418,800(8.4)	421,133(8.4)	395,000(7.9)	375,100(7.5)	240,000(4.8)
Sony	177,900(3.6)	220,300(4.4)	112,100(2.2)	171,235(3.4)	69,067(1.4)	150,000(3.0)
Tokyo Gas	792,333(15.8)	736,133(14.7)	622,300(12.4)	554,550(11.1)	216,867(4.3)	372,000(7.4)
KDDI	374,500(7.5)	316,233(6.3)	208,633(4.2)	-97,000(-1.9)	-107,000(-2.1)	-576,000(-11.5)
Nomura Holdings, Inc.	684,400(13.7)	538,933(10.8)	578,900(11.6)	178,050(3.6)	539,400(10.8)	-985,500(-19.7)
Shin-Etsu Chemical	139,166(2.8)	133,800(2.7)	179,200(3.6)	112,050(2.2)	-171,500(-3.4)	-264,000(-5.3)
Nippon Steel Cooperation	455,633(9.1)	404,300(8.1)	464,566(9.3)	395,750(7.9)	256,833(5.1)	399,000(8.0)
Shiseido	576,833(9.5)	403,366(8.1)	430,100(8.6)	550,850(11.0)	416,767(8.3)	510,000(10.2)
Sumitomo-bank	133,766(2.7)	202,766(4.1)	141,500(2.8)	161,135(3.2)	82,300(1.6)	64,000(1.3)
Kyosera	331,666(4.6)	257,166(5.1)	208,000(4.2)	305,750(6.1)	222,533(4.5)	208,000(4.2)
East Japan Railway	175,733(1.5)	157,166(3.1)	6,566(0.1)	245,500(4.9)	135,033(2.7)	45,000(0.9)
Sekisui House, Ltd.	293,566(5.9)	314,966(6.3)	390,333(7.8)	407,235(8.1)	256,433(5.1)	232,000(4.6)
Tokyo Electric Power	121,000(2.4)	75,466(1.5)	-26,900(-0.5)	17,550(0.4)	-393,000(-7.9)	-27,000(0.5)
Daiwa House Industry	59,200(1.2)	116,100(2.3)	106,500(2.1)	150,350(3.0)	-176,033(-3.5)	54,100(1.1)
Ajinomoto	691,100(13.8)	496,733(9.9)	438,666(8.8)	370,500(7.4)	507,200(10.1)	401,000(8.0)
Average	363,913(7.3)	306,856(6.1)	270,311(5.4)	264,228(5.3)	173,890(3.5)	43,952(0.9)

Chapter 4

A Portfolio Optimization Model using Genetic Network Programming with Control Nodes

4.1 Introduction

This chapter presents an application of Genetic Network Programming with control nodes (GNPcn) to the problem of multi-brands optimization, which is one kind of portfolio optimization. Since the portfolio optimization in the stock market consists of deciding what brands to include in a portfolio given the investor's objectives and economic conditions, the always difficult selection process includes identifying which brands to purchase, how much, and when. The basic idea is that we want to choose a group of brands from a large number of available issues, in order to maximize the expected return given an acceptable risk rate. A rational investor needs to consider not only maximizing the profit of the investment, but also minimizing the uncertainty or risk resulting from the fluctuations that are expected in the portfolio. The main problem to be solved in this chapter is how to allocate the available capital to different brands in a portfolio in order to maximize profit and minimize risk simultaneously.

Base on our previous study on GNP, Genetic Network Programming with control nodes [62], [63] has been proposed in order to extend the functions of conventional GNP. Since GNP has a directed graph structure, the aim of GNPcn is to improve the performance by extending the evolutionary method of it. In traditional GNP, the current node isn't compulsorily transferred to the start node. However, in the GNPcn method, the number of control nodes is set up and a certain number of processing

nodes are executed before returning to one of the control nodes, i.e., we extend the breadth and depth of searching space for GNP. It is clarified from the simulations that the performance of GNP could be improved by the combination with control nodes.

In this chapter, we applied GNPcn to the financial field in order to create an efficient portfolio optimization model for given multi-brands. The features of the proposed method compared with other traditional methods are as follows: The GNPcn method makes a stock trading strategy considering the recommendable information of technical indices and candlestick charts [39] for efficient trading decision making. Reinforcement Learning is also used in this chapter for taking appropriate actions.

Section 4.2 presents the literature review. Section 4.3 describes the proposed GNPcn approach to be studied in this chapter. In Section 4.4, we explain the optimization algorithm in brief. Section 4.5 presents experimental environments, conditions and results using GNPcn method. The trading profits are presented and compared with both the traditional GNP method and Buy&Hold method. Finally, Section 4.6 concludes this chapter.

4.2 Literature Review

In recent decades, the portfolio problem in financial engineering has received a lot of attention. The foundation of portfolio optimization was laid by Harry Markowitz [64]-[67], where he proposed a mean-variance optimization model for designing an optimum portfolio based on the idea of minimizing risk and maximizing expected returns. As we know, it is difficult to acquire good estimates for the expected returns, and the calculation of risk becomes more and more complex as the number of available assets grow. In the case of linear constraints, the problem can be solved efficiently by parametric quadratic programming. However, there are many real-world non linear constraints which limit the number of different assets in a portfolio. As a consequence, evolutionary computation was developed to calculate the optimal portfolio since portfolio problems make the search space become larger. In this paper, how to allocate the given money to a certain number of fixed brands is discussed, which is different from the classical mean-variance optimization model.

Over the last few decades, various approaches have been applied to several financial problems, especially for stock market activities. Generally speaking, these ap-

proaches can be separated into two categories: statistical and AI. The statistical methods are widely used to predict the stocks based on the past time series data. The traditional statistical approaches include ARMA method [68], the threshold Autoregressive model [69], Smooth Transition Autoregressive model(STAR) [70], the Autoregressive Conditional Heteroscedastic (ARCH) [71] and multiple Linear Regression model. These methods rely on the assumption of linearity among variables and normal distribution. However, with statistical models, problems arise when the variance in the time series increase or when nonlinear processes exist in the time series. On the other hand, AI approaches, with the increasing need for more efficient trading models in the stock market, has been confirmed to outperform the conventional statistical models for that it overcomes the limitation of such an assumption [72].

As a main approach in the AI field, Artificial Neural Networks (ANNs) has been widely used for its ability to forecast financial performances. Dropsy [73] uses ANNs as a nonlinear prediction tool to forecast international equity risk, in which both linear and nonlinear forecasting results outperform the random work. Lam [74] applied the back-propagation algorithm to integrate fundamental and technical analysis for financial performance prediction. The experimental results show that the ANNs outperform the benchmark. Both the forecasting and decision models are significantly outperforming the benchmark market performance. Likewise, Yu et al. [75] introduce a new Artificial Intelligence technique and propose a fast and efficient radial basis function (RBF) neural network-based methodology. However, despite the wide spread use of ANNs in the financial domain, there are significant problems to be addressed. Since ANNs are data-driven model, the underlying rules in the data are not always apparent, which leads to so-called black-box models, and investors cannot benefit from the knowledge discovery in the analytic process. In addition, the buried noise and complex dimensionality of the stock market data make it difficult to learn or re-estimate the ANNs parameters [76].

Genetic Algorithms (GA), as one of the most popular heuristic optimization techniques, were originally developed by Holland [1]. Subsequently, GA had been applied to many optimization problems in engineering and operations research. GA is a good tool for optimization problems since they make no restrictive assumptions about the solution space. Lin considered the multi-objective genetic algorithm for portfo-

lio selection problem [77]. Oh proposed a new portfolio selection algorithm based on portfolio beta by using genetic algorithm [35]. Moreover, Lin and Liu [78] considered Markowitz's model with minimum transaction lots and they presented three other models using GA as their solver. Deck used GA to train a neural network trading system. However, when GA was applied to the portfolio optimization, the problem is that many chromosomes are coded into the same portfolio, or similar chromosomes may be coded into very different portfolios which makes it more difficult for GA to produce better chromosomes from good ones. These problems multiply the GA's search space and makes GA less efficient in finding the optimal portfolio.

Genetic Programming (GP), which has been described by Koza (1992) [6], can be considered as an extension of GA. It uses tree-like individuals that can represent mathematical expressions. So far, GP has been applied to wide range of financial fields such as stock trading system [34], bankruptcy prediction [79], and etc. In comparison with GA, GP allows the optimization of much more complicated structures and can therefore be applied to a greater diversity of problems [80]. Recent applications of GP has been done, when the notion of risk is not considered [81]. Although GP are widely used in the financial field, it occasionally causes some bloating problems for its tree structure.

As another tool based on expert knowledge, Fuzzy Logic can be used for stock market forecasting domain either independently or hybridized with other methods. For instance, Romahi and Shen [82] developed an evolving rule based expert system to forecast the financial market activity, where the fuzzy logic and rule induction were combined together to obtain a promising method. However, fuzzy logic still has its limitation when applied to the stock market. That is, when designing the fuzzy models for forecasting or trading, we need to get the expert knowledge prior to it.

Though many AI approaches have been applied to the financial field, due to such kinds of bottlenecks, we proposed GNP and GNP-RL as an extended method of GA and GP and its effectiveness has been confirmed in our former studies. In this chapter, GNP with control node (GNPcn) has been firstly applied to the multi-brands optimization model for the stock trading.

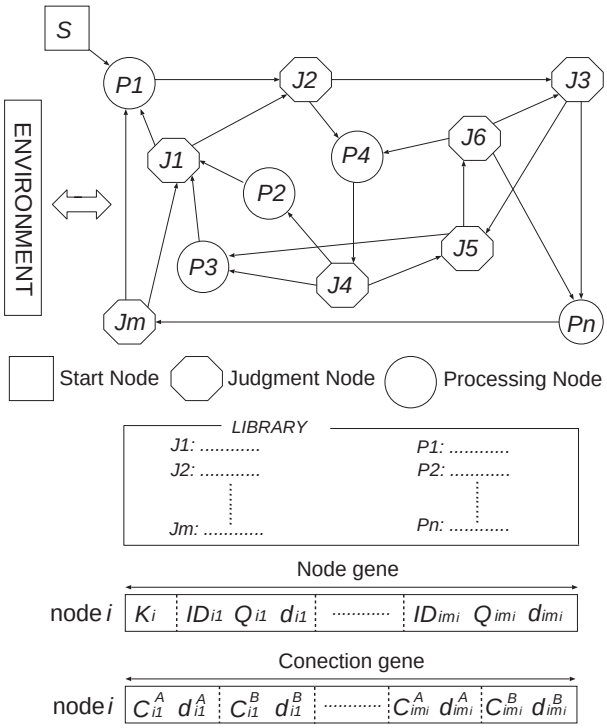


Figure 4.1: The basic structure of GNP

4.3 Genetic Network Programming with Control Nodes

In this section, Genetic Network Programming (GNP) with Control Node is explained briefly. Basically, GNP is an extension of GP in terms of gene structures. The original idea is based on the more general representation ability of directed graphs than that of trees. The graph structure of GNP has some inherent characteristics such as compact structures and an implicit memory function that contribute to creating effective action rules.

4.3.1 Conventional Genetic Network Programming

The detailed explanation for conventional GNP can be found in Chapter 2, and Fig. 4.1 shows a basic structure of GNP. The genotype expression of GNP node is also shown in Fig. 4.1. K_i represents the node type, and ID_i represents an identification number of the node function. Q_{ip} means Q value which is assigned to each state and action pair. d_{ip} ($1 \leq p \leq m_i$, m_i is the number of subnodes in judgment and processing nodes) is the time delay spent on the judgment or processing at node i, while d_{ip}^A, d_{ip}^B, ... are time delays

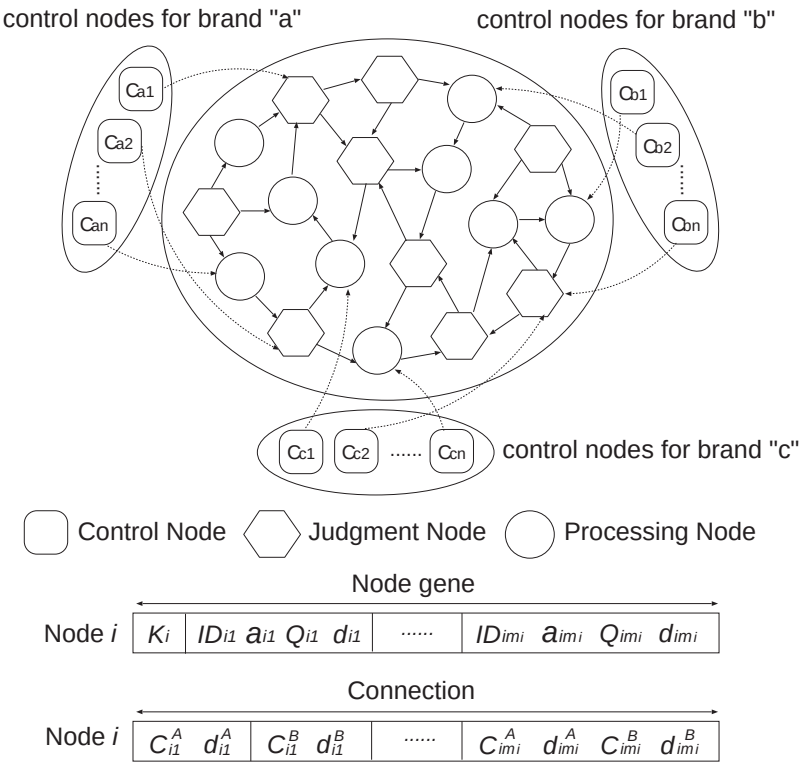


Figure 4.2: The basic structure of GNPcn

spent on the node transition from node i to the next node. C_{ip}^{A}, C_{ip}^{B}, ... show the node number of the next node.

Especially, in conventional GNP, the current node isn't compulsorily transferred to the start node and there is no terminal node in GNP. Therefore, once GNP is booted up, the successive activation of GNP system is carried out according to the network flow until the time limit.

4.3.2 Genetic Network Programming with Control Nodes (GNPcn)

In conventional GNP, since the current node isn't compulsorily transferred to the start node, there is a possibility that some of the nodes are not used. Therefore, in this chapter, GNP with control nodes (GNPcn) is used (see Fig. 4.2) to solve the multi-brands problem. GNPcn starts from a control node, and the current node is transferred back to one of the control nodes after executing a certain number of processing nodes. Consequently, the performance of GNPcn improves because the increase of the number of control nodes contributes to searching a solution space widely and finding many dis-

tinguished trading rules. Especially, the number of the control nodes can be considered as breadth of the search of GNP, while the number of processing nodes activated per control node can be considered as depth of the search of GNP.

In addition, the design on what kinds of branches the judgment nodes should have is so important similarly to the design of the processing nodes in GNP. In this case, the transition or the behavior of GNP is mainly determined by which branch the judgment node selects. Therefore, GNP should have the possibility for selecting many judgment results in GNP. In order to overcome such a problem, in this chapter, GNPcn is proposed and basic studies are done where the number of control nodes is fixed and the number of processing nodes activated per control node is evolved.

Fig. 4.2 shows the basic structure of GNPcn. GNPcn has several control nodes, judgment nodes and processing nodes. GNPcn uses one of the groups of control nodes for one brand of stocks, so that GNPcn could deal with multi brands. When GNPcn deals with brand "a", GNPcn starts its node transition from control node "C_{a1}", and the current activated node returns to one of the control nodes (C_{a1}, ..., C_{an}) successively after transiting m processing nodes from the last control node. This procedure is as follows:

- First, the activated node is determined by the control node of each brand.
- Execute the judgment or processing function of the current activated node of the brand, then determines the next node which is executed in the next time period.
- The next node is determined by the next control node (such as C_{a1}, ..., C_{an}) after m processing nodes are executed from the last control node.

4.3.3 Genetic Operations of GNPcn

Fig. 4.3 shows the whole flowchart of GNPcn. In this sub-section, the genetic operators in the evolution phase are introduced. GNPcn with RL has three kinds of genetic operators: selection, crossover and mutation. In GNPcn, mutation operation could be executed not only on the connections between nodes but also on the node functions and the number of processing nodes activated per control node.

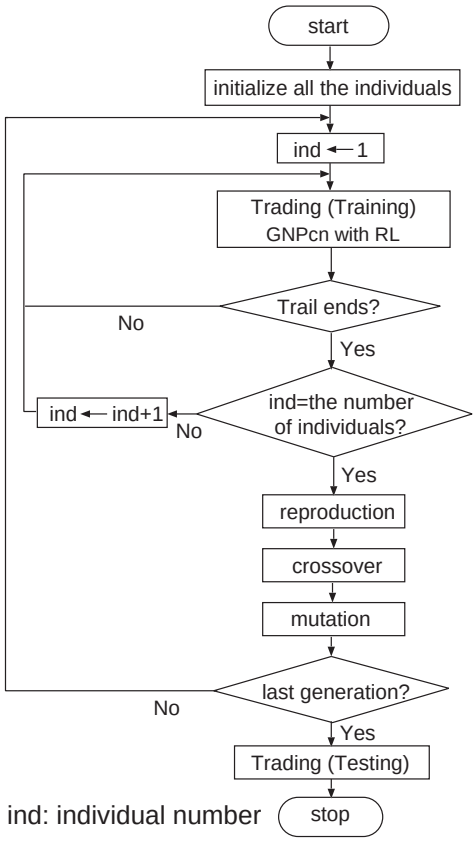


Figure 4.3: Flowchart of GNPcn

Crossover

Crossover is executed between two parents and generates two offspring. The procedure of crossover is as follows.

(1) Select two individuals using tournament selection twice and produce them as parents.

(2) Each node is selected as a crossover node with the probability of P_c.

(3) Two parents exchange the genes of the corresponding crossover nodes.

(4) Generated new individuals become the new ones of the next generation.

If GNP exchanges the genes of judgment nodes, it must exchange all the genes with suffix $A, B, C, \ldots$ simultaneously.

Mutation

Mutation is executed in one individual and a new one is generated. The procedure of mutation is as follows.
(1) Select one individual as a parent using tournament selection.
(2) Mutation operation

- change connection: Each node branch (C_{ip}^{A}, C_{ip}^{B},...) is selected with the probability of P_m, and the selected branch is reconnected to another node.
- change node function: Each node function (ID_{ip}) is selected with the probability of P_m, and the selected function is changed to another one.
- change node number: The number of processing nodes activated per control node (m) is changed to the other value with the probability of P_m.

(3) Generated new individual becomes the new one of the next generation.

4.3.4 Learning Phase of GNPcn

In this chapter, we use GNPcn with Sarsa learning which is one kind of reinforcement learning methods. Sarsa can obtain Q values which estimate the sum of the discounted rewards obtained in the future. Suppose an agent selects an action a_t at state s_t at time t, a reward r_t is obtained and an action a_{t+1} is taken at the next state s_{t+1}. Then $Q(s_t, a_t)$ is updated as follows.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

α is a step size parameter, and γ is a discount rate. The state means the current node and action means the selection of the sub-node in this chapter.

4.4 Portfolio Optimization Algorithm using GNPcn

4.4.1 Technical Indices and Candlestick Chart

In the proposed portfolio optimization model, technical indices and candlestick chart are used as judgment functions. Concretely speaking, each judgment node uses one of the following technical indices for its judgment: Rate of Deviation from moving average (ROD), Relative Strength Index (RSI), Rate of Change (ROC), Volume Ratio,

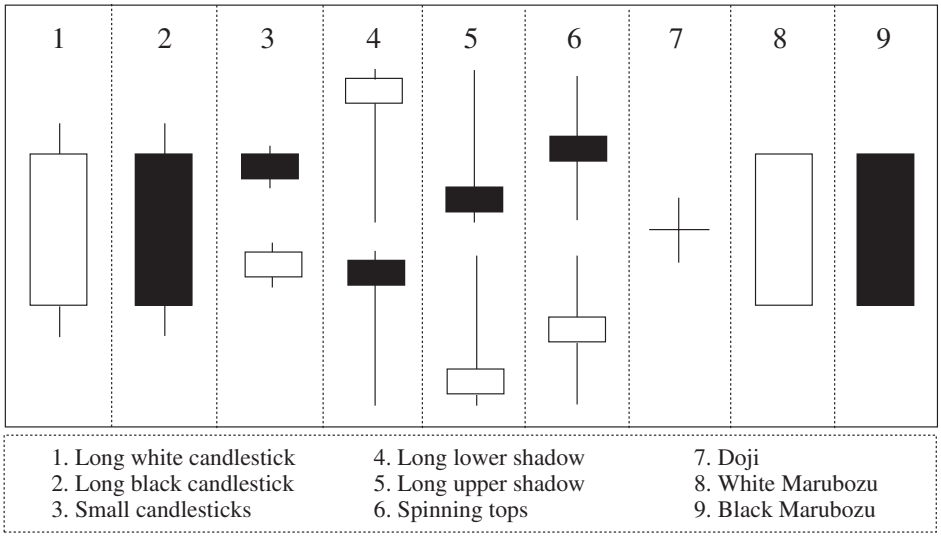


Figure 4.4: Candlestick chart patterns

Rank Correlation Index (RCI), Stochastics, Golden/Dead cross and Moving Average Convergence and Divergence (MACD).

As a judgment function, a candlestick chart was also used in order to elicit technical knowledge. The candlestick chart is a useful tool to visualize the stock prices so that investors can detect patterns which can be used to predict future stock price movements. As illustrated in Fig. 4.4, the candlestick chart consists of a rectangle and two shadow lines. The rectangle which is called the real body indicates the difference between the opening value and closing value of the stock. If the real body of the candlestick chart shows that the opening value is higher than the closing value, the candlestick chart is called black candlestick. On the other hand, if the closing value is higher than the opening value, the candlestick chart is called white candlestick chart. The white candlestick implies the rising signal of the stock price and the black candlestick implies the falling signal. The stock price patterns which are represented by the candlestick chart shapes give important clues to predict future stock price movements. Thus, the technical knowledge from the candlestick chart was used as a key information for predicting stock price movements in this study.

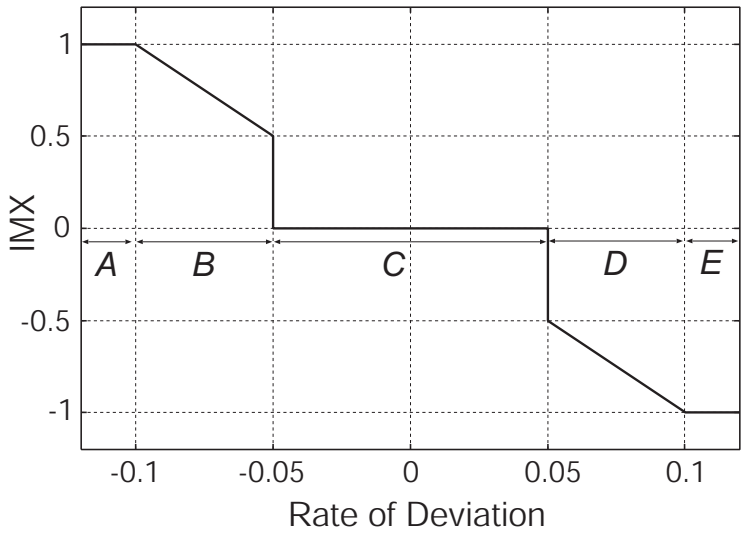


Figure 4.5: IMX functions in judgment nodes (In case of ROD)

Table 4.1: Functions of control nodes, judgment nodes and processing nodes

0	control node
1	judgment node to check the technical indices
1	judgment node to check the candlestick charts
2	processing node for buying action
2	processing node for selling action

4.4.2 Judgment Function of GNPcn

Table 4.1 shows the node functions of GNPcn. When a current activated node i is a judgment node, first, one Q value is selected from Q_{i1}, ..., Q_{im_i} based on ϵ-greedy policy (m_i is the number of subnodes in judgment and processing nodes). Then, the corresponding function (ID_{ip}) is selected. The gene ID_{ip} shows a technical index or a candlestick GNP judges at node i.

Each technical index has its own judgmment function according to the technical value as shown in Fig. 4.5, where x axis shows the value of each technical index, and the y axis shows the IMX output (IMX is the Importance Index which has been proposed in our former research [18]) of the judgment function. The shape of IMX functions is fixed in our study according to the knowledge from the real stock market. A, B, C, ... correspond to judgment results. Suppose Q_{i1} and the corresponding ID_{i1}=1 are

selected (judgment of rate of deviation), and if the rate is more than 0.1, the judgment result becomes E, and the next node number becomes C_{i1}^{E}.

As shown in Fig. 4.4, the candlestick chart is used as one of the judgment functions in this chapter. The proposed method has judgment nodes which check candlestick chart patterns. The judgment function of the candlestick chart is executed as follows. When the selected sub-node has a judgment function of the candlestick chart, GNP judges the current day's candlestick by the price information. In our study, we use nine typical candlestick chart patterns as shown in Fig. 4.4, and following is a brief introduction about these patterns:

- 1. Long white candlestick: It indicates the signal of uptrend movement. (the longer the body, the more significant the price increase)
- 2. Long black candlestick: It indicates the signal of downtrend movement. (the longer the body, the more significant the price decrease)
- 3. Small candlesticks: It indicates that neither bullish nor bearish signal could control the movement and prices finished about where they started.
- 4. Long lower shadow: It indicates bullish signal.(the longer the lower wick, the more reliable the signal)
- 5. Long upper shadow: It indicates bearish signal.(the longer the upper wick, the more reliable the signal)
- 6. Spinning tops: It indicates a standoff of bullish and bearish signal, which are neutral patterns.
- 7. Doji: It indicates a standoff, and a turning point could be developing.
- 8. White Marubozu: It indicates dominant bullish trades with continued bullish trend.
- 9. Black Marubozu: It indicates dominant bearish trades with continued bearish trend.

As an example, when the candlestick pattern is "3", GNPcn selects third branch to transfer to the next node.

4.4.3 Processing Function of GNPcn

When the current node is a processing node, the following will be carried out:

1) First, one Q value is selected from Q_{i1}, ..., Q_{im_i} based on ϵ-greedy policy. That is, a maximum Q value among Q_{i1}, ..., Q_{im_i} is selected with the probability of 1-ϵ, or a random one is selected with the probability of ϵ. Then, the corresponding ID_{ip} is selected.

2) Determine buying or selling:

- In the case of ID_{ip}=0 (buy): GNPcn buys the stock of brand $b \in B$ as much as possible using $Initial(V, b)$, where $Initial(V, b)$ is the initial budget of brand b used for validation. It is explained in the next subsection.
- In the case of ID_{ip}=1 (sell): GNP sells all of the stocks of brand $b \in B$ in hand.

3) If ID_{ip} is selected, the next node number becomes C_{ip}^{A} when the current node is transferred to the next node.

4.4.4 Multi-Brands Optimization Algorithm

In this section, how to determine the initial budget of each brand is described in the dealing of stocks using GNP with control nodes. This portfolio optimization system has been constructed by training phase and validation phase (see Fig. 4.6). Especially, the portfolio does not change during the validation period, although it changes generation by generation in the training period.

Notations

- $Initial(t)$: initial budget for training
- $Initial(v)$: initial budget for validation
- $Initial(t, b, n)$: initial budget of brand b in the nth generation for training
- $Initial(v, b)$: initial budget of brand b for validation
- T: temperature parameter
- B: set of brands

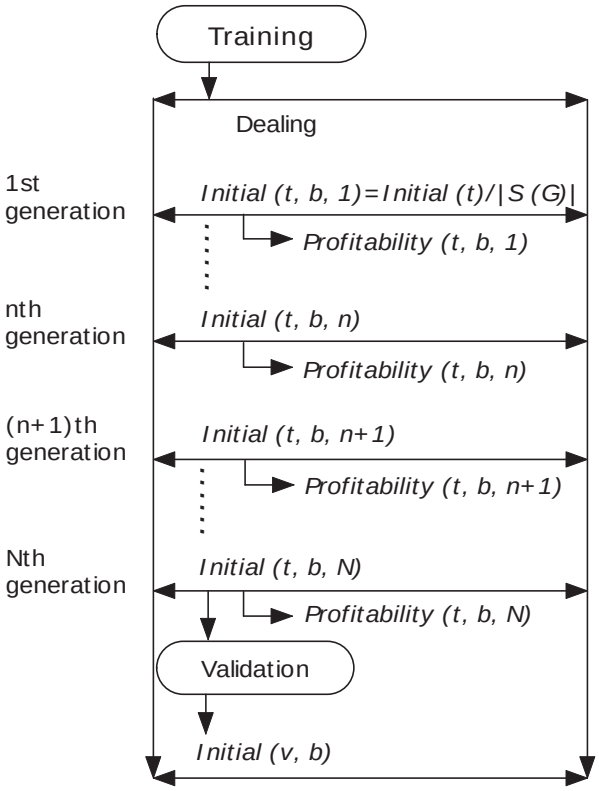


Figure 4.6: Training and validation phase

- b: brand index
- N: number of generations
- n: generation number
- $Sell(t, b, n)$: sum of the money obtained by selling the stocks of brand b during the dealing in the nth generation for training
- $Buy(t, b, n)$: sum of the money paid by buying the stocks of brand b during the dealing in the nth generation for training
- $Profit(t, b, n)$: money gained or lost during the dealing of brand b in the nth generation for training, where $Profit(t, b, n) = Sell(t, b, n) - Buy(t, b, n)$
- $Profitability(t, b, n)$: profitability during the dealing of brand b in the nth generation for training, where $Profitability(t, b, n) = Profit(t, b, n)/Initial(t, b, n)$

- $Profit(v, b, d)$: money gained or lost during the dealing of brand b until day d for validation

We use GNPcn for determining the portfolio of the stock brands, where each brand corresponds to each group of control nodes like shown in Fig. 4.2. In GNPcn, judgment nodes check the technical indices and candlestick chart patterns, and processing nodes work for buying or selling stocks.

One of the groups of control nodes is assigned to each brand $b \in B$. The current activated node returns to one of the control nodes of each brand after transiting m processing nodes from the last control node. That is, the current activated node returns to C_{b2} firstly, and next C_{b3}, C_{b4}, ..., C_{bn}. Trading timing is determined by the GNPcn evolution.

The fitness function of the GNPcn in the *nth* generation is defined as follows,

$$Fitness(n) = \sum_{b \in B} Profit(t, b, n).$$

In the following, how to determine the portfolio is explained.

Training Phase

The initial budget $Initial(t, b, 1)$ of brand b in the first generation is calculated by Eq. (1).

$$Initial(t, b, 1) = \frac{Initial(t)}{|B|}. \tag{4.1}$$

After the first generation, we can get the following profit and profitability.

$$Profit(t, b, 1) = Sell(t, b, 1) - Buy(t, b, 1).$$

$$Profitability(t, b, 1) = \frac{Profit(t, b, 1)}{Initial(t, b, 1)}.$$

Then, the initial budget $Initial(t, b, 2)$ of brand b in the second generation can be calculated by Eq. (2) considering the profitability gained in the first generation.

$$Initial(t, b, 2) = \frac{exp(Profitability(t, b, 1)/T)}{\sum_{b \in B} exp(Profitability(t, b, 1)/T)} Initial(t). \tag{4.2}$$

Likewise, the initial budget $Initial(t, b, n+1)$ of brand b in the $(n+1)th$ generation can be calculated by Eq. (3) considering the profitability gained in the *nth* generation.

$$Initial(t, b, n+1) = \frac{exp(Profitability(t, b, n)/T)}{\sum_{b \in B} exp(Profitability(t, b, n)/T)} Initial(t), \tag{4.3}$$

where,

$$Profit(t,b,n) = Sell(t,b,n) - Buy(t,b,n),$$

$$Profitability(t,b,n) = \frac{Profit(t,b,n)}{Initial(t,b,n)},$$

Validation Phase

The initial budget $Initial(v,b)$ of brand b in the validation phase is given as follows.

$$Initial(v,b) = \frac{exp(Profitability(t,b,N)/T)}{\sum_{b \in B} exp(Profitability(t,b,N)/T)} Initial(v), \tag{4.4}$$

where, $Initial(v)$ is the initial budget for the validation.

GNPcn individual starts its operation from one of the control nodes in the validation and the activated node is transferred to a judgment node or a processing node. At the processing node, the trading is executed using the opening price of the day. The concrete procedure of the trading is as follows. Do the following for each brand until the end of the trading:

- If the current node is a judgment node, it determines the next node depending on the judgment result. If the current node is a processing node, it transits to the next node after buying or selling stocks.
- Calculate the available budget of each brand whenever the buying signal occurs in the processing nodes during the transition of each brand.
- When the processing node is executed m times from the last control node, the next node is determined by the next control node.
- Calculate the profit and profitability of each brand at the end of the trading.

4.5 Simulations

To confirm the effectiveness of GNPcn for the portfolio optimization system, we carried out the trading simulations using 10 brands selected from the companies listed in the first section of Tokyo stock market in Japan (see Table 4.2). The simulation period is divided into two periods: one is used for training and the other is used for testing.

- Training: January 4, 2001—December 30, 2003 (737 days)
- Testing: January 5, 2004—December 30, 2004 (246 days)

We suppose that the initial funds, i.e., $Initial(t)$ and $Initial(v)$ are 50,000,000 Japanese yen in both periods.

Especially, there is an advantage of GNPcn over the conventional GNP: In the simulations, GNPcn is compared with the traditional GNP in which one GNP individual deals with only one brand, so we evolved 10 different GNP populations. When we use GNPcn, one GNPcn has 10 groups of control nodes, each of which deals with one brand, so one GNPcn can deal with 10 brands.

4.5.1 Settings of GNPcn

Table 4.3 shows the parameters of the evolution of GNPcn. GNPcn uses judgment nodes which judge the technical indices and candlestick charts. The technical indices are calculated using three kinds of calculation periods (short-term, medium-term and long-term) except Golden/Dead cross and MACD. Therefore, the number of kinds of judgment nodes is 21 (including one candlestick judgment). The number of processing functions is two: buying and selling. Five control nodes are assigned to each brand and we tested 10 brands, i.e., the total number of control nodes is 50. The total number of nodes in each individual is 80 including 10 processing nodes, 20 judgment nodes and 50 control nodes. However, the functions ID_{ip} in sub-nodes are determined randomly at the beginning of the first generation, and changed appropriately by evolution.

The initial connections between nodes are also determined randomly at the first generation. At the end of each generation, 179 new individuals are produced by mutation, 120 new individuals are produced by crossover, and the best individual is preserved. The other parameters are the ones showing good results in the simulations. The initial Q values are set at zero.

4.5.2 Simulation Results

In Table 4.4, we test the various values of T in Eq. (2), which is called temperature parameter. As we know, according to the Boltzman distribution theory, different T value defines different strength of distribution. When it is applied to this portfolio

Table 4.2: Profits in the testing simulations {Profit[yen](profitability[%])}

Brand	GNPcn	GNP-RL	GNP-Candlestick	GA	Buy&Hold
NEC Corporation	—	43,400(0.9)	-126,150(-2.5)	-531,000(-10.6)	-1,026,000(-20.5)
Fuji Heavy Ind.	—	188,800(3.8)	97,700(2.0)	173, 000(3.5)	-189,000(-3.8)
East Japan Railway	—	418,800(8.4)	96,550(1.9)	7,590(0.2)	477,000(9.5)
KDDI	—	316,233(6.3)	-76,600(-1.5)	-273,000(-5.5)	-576,000(-11.5)
Nomura Holdings, Inc.	—	638,933(12.8)	-293,785(-5.9)	-374,087(-7.5)	-985,500(-19.7)
Shin-Etsu Chemical	—	133,800(2.7)	7,250(0.1)	-539,400(-10.8)	-264,000(-5.3)
Sony	—	202,766(4.1)	280,500(5.6)	112,000(2.2)	150,000(3.0)
Tokyo Electric Power	—	116,100(2.3)	-65,500(-1.3)	360,750(7.2)	262,500(5.3)
Hitachi	—	403,366(8.1)	472,300(9.4)	276,300(5.5)	336,000(6.7)
Nissan	—	464,966(9.3)	590,750(11.8)	372,000(7.4)	450,000(9.0)
Average	4,262,714(8.5)	292,716(5.9)	98,302(2.0)	-41,585(-0.8)	-136,500(-2.7)

Table 4.3: Parameter conditions for evolving GNP

Number of individuals=300 (mutation:179, crossover:120, elite:1)
Number of nodes=80 (Judgement node=20, Processing node=10, control node=50)
Number of sub-node in each node=2
P_c=0.1, P_m=0.03, α=0.1, γ=0.3, ϵ=0.1

study, T value indicates the initial budget of each brand distributed by GNPcn. If $T \rightarrow \infty$, GNPcn distributes the initial budget to each stock evenly. On the other hand, if $T \rightarrow 0$, all the money is distributed to the stock brand which obtain the best profitability.

Table 4.4 shows the average fitness values over 30 independent simulations in the training when the parameter T is set to different values. From the results, it is clarified that the we can get a good fitness value in the training period when T is set to 0.003. Therefore, we set the value of 0.003 for the temperature parameter T in our simulations.

Fig. 4.7 shows the fitness and profit curves, i.e., $Fitness(n)$ and $Profit(t, b, n)$ of each brand $b \in B$ in the nth generation using the data from 2001 to 2003, and the line is the average value over 30 independent simulations. From Fig. 4.7, we can see that GNPcn with reinforcement learning can obtain larger profits for the training data as the generation goes on.

Fig. 4.8 shows the initial budget change of the 10 brands, i.e., $Initial(t, b, n)$ in the training period, while Fig. 4.9 shows the profitability change of each brand, i.e., $Profitability(t, b, n)$ in the training period. From these figures, it is clear that the brand which can obtain larger profitability can also get the initial budget more than other

Table 4.4: Average fitness values [million yen] in training period with different T

T	0.001	0.003	0.005	0.01	0.03	0.05
Fitness	133.5	143.2	136.2	125.6	110.1	108.2

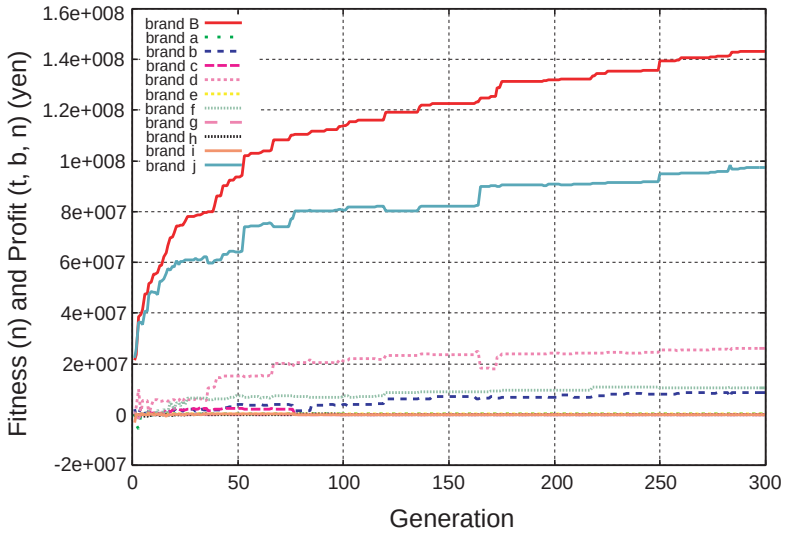


Figure 4.7: Fitness and profit curves of 10 brands in the training period

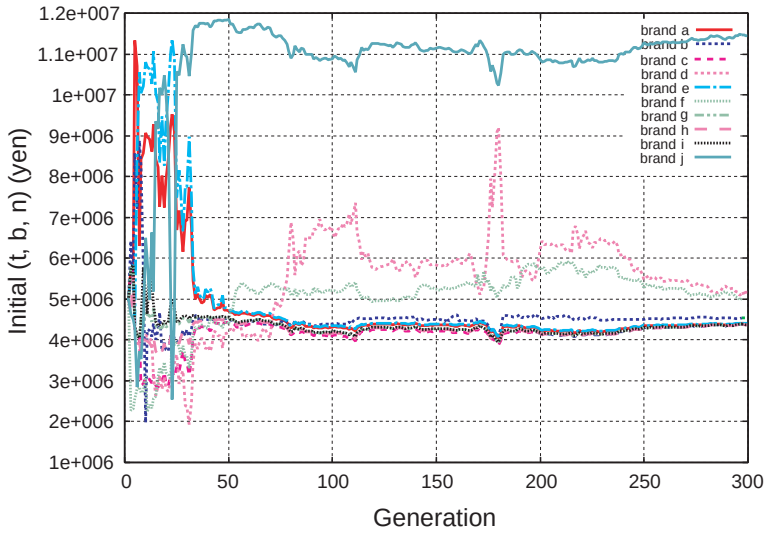


Figure 4.8: Initial budget change of 10 brands in the training period

brands. The optimization of the proposed portfolio model has been shown consequently.

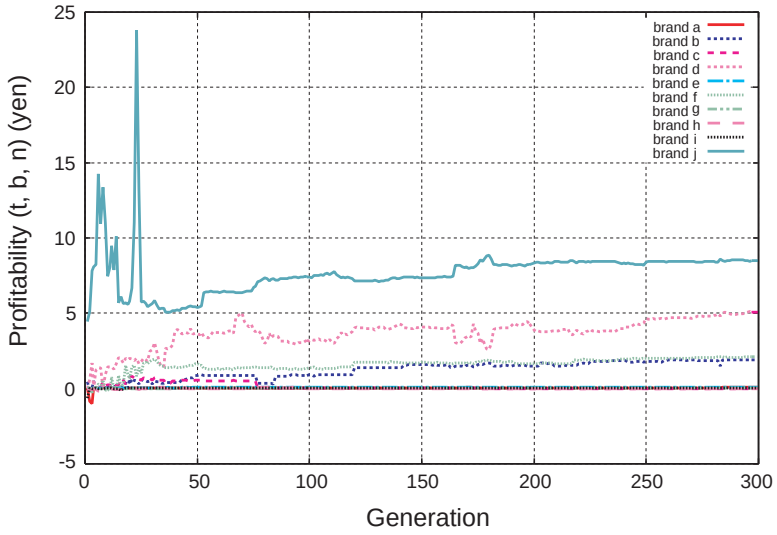


Figure 4.9: Profitability change of 10 brands in the training period

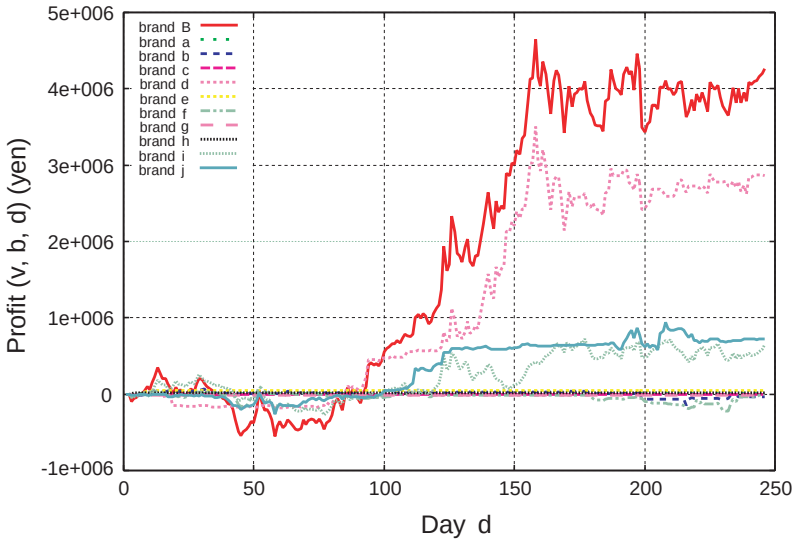


Figure 4.10: Profits change of 10 brands in the testing period

Table 4.2 shows the profit and profitability in the testing term using the data in 2004. The values in Table 4.2 are the average profit and profitability of 30 independent simulations with different random seeds. For the comparison, Table 4.2 also shows the results of GNP-RL, GNP-Candlestick [39], GA and Buy&Hold method which is often considered as a benchmark in trading stocks simulations. Buy&Hold buys as much stocks as possible at the opening of the market on the first day in the simulations,

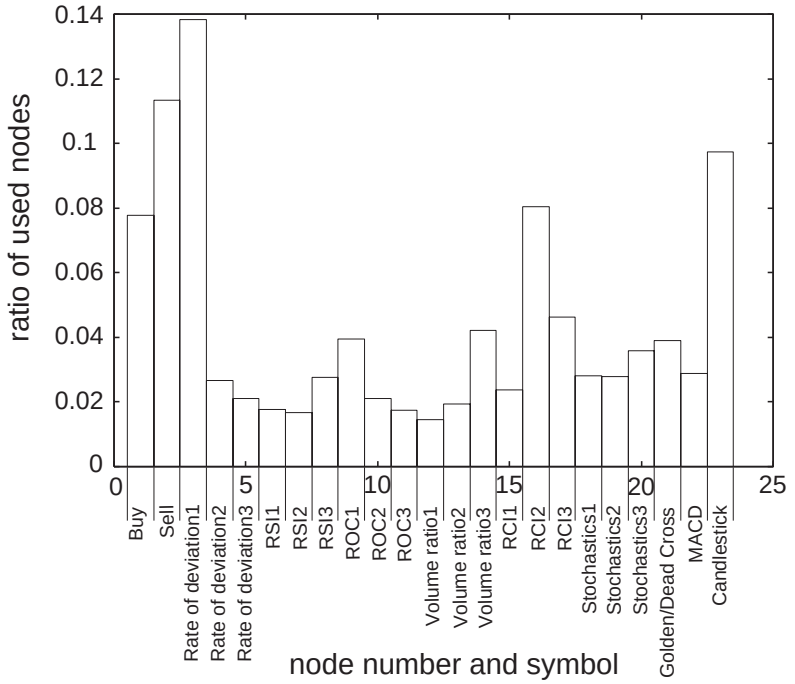


Figure 4.11: Ratio of nodes used by GNPcn in the testing period

and sells all the stocks at the opening on the last day. From Table 4.2, the proposed method can obtain larger profits than Buy&Hold and all the other traditional methods in most of the trade in 10 brands. Since the initial budget of the traditional methods and Buy&Hold are 5,000,000 yen respectively, we set the initial budget of the proposed model at 50,000,000 yen. This is because GNPcn deals with all of the 10 brands simultaneously, while in the case of conventional GNP method, one GNP deals with only one brand. In order to compare these methods under the same condition, we set the initial budget of GNPcn at 10 times of other methods. As a result, the average profitability can be calculated and compared with each other. Especially, the profit for each brand can be calculated in the traditional methods, however, in the proposed model, GNPcn deals with all of the brands, so, only the total profit is described.

Fig. 4.10 shows the $Profit(v, b, d)$ during the dealing of brand $b \in B$ until day d using the data of 2004. From Fig. 4.10, we can see that the profit keep increasing during the testing period. The advantage of the proposed method is to determine the distribution of the initial budget to each brand automatically, and the brands which obtain larger profitability can have the initial budget more than other brands. As a result, by this

efficient portfolio optimization system, GNPcn could obtain much profit in the trading of those brands.

Fig. 4.11 shows the average ratio of the nodes used in the test period over 30 independent simulations in order to see which nodes are used and which are most efficient for stock trading model. The total number of node functions is 23, while each processing node has a node number of (1-2), and each judgment node has a node number of (3-23). The x-axis shows the kinds of nodes, while the y-axis shows the average ratio of the used nodes. From the figure, we can see that the processing nodes are used to determine buying and selling stocks, and the judgment nodes of "Rate of Deviation 1" and "Candlestick" are frequently used. Thus, it can be said that GNPcn judges that these node functions are important to determine stock trading. GNPcn can automatically determine which nodes should be used in the current situation by evolving node functions and connections between nodes, in other words, GNPcn can optimize the combination of technical indices and candlestick charts used for this multi-brands stock trading model.

4.6 Summary

In this chapter, we proposed a multi-brands optimization algorithm by using GNPcn to check the information of technical indices and candlestick chart patterns. Compared to conventional GNP, GNPcn has many control nodes, and each group of control nodes is assigned to each stock brand for creating effective portfolio strategy. Since GNPcn can adjust the distribution of the initial budget to each brand at each generation and more budget is assigned to the brands with larger profitability, we can create the effective portfolio optimization system and get more profit. We carried out the simulation of GNPcn method using stock data of 10 brands for four years. In the simulations, GNPcn is compared with several conventional methods and Buy&Hold. From the results, it is clarified that GNPcn performs much better in terms of the profits than traditional GNP methods, GA and Buy&Hold method. It shows the efficiency of the proposed GNPcn method for dealing with the portfolio optimization problem.

We still have some further study in the future. First, the algorithm presented can be further improved by modifying evolutionary operators, especially mutations. The fitness function can also be improved to increase the efficiency of the method. More-

over, additional efforts should be spent on the methods of portfolio selection in order to eliminate unacceptable solutions and get the best combination of stock brands. We will also evaluate the proposed method comparing with other methods in the financial market.

Chapter 5

A Portfolio Selection Model using Genetic Relation Algorithm and Genetic Network Programming

5.1 Introduction

This chapter presents an application of the evolutionary computation methods named genetic relation algorithm (GRA) and genetic network programming (GNP) to the problem of portfolio selection in the financial field. In the conventional portfolio optimization problem, given the investor's objectives and economic conditions, we want to find out what assets to include in an optimal portfolio, in order to maximize the expected return and minimize risk simultaneously.

In order to solve this optimization problem, Harry Markowitz [64]-[67] first proposed a mean-variance optimization model to design an optimum portfolio as the foundation of portfolio selection, which assumes that the total portfolio can be obtained using the mean return of the assets and the variance of the return over these assets. In the case of linear constraints, Best and Kale [83] and Stein et al. [84] solved the problem efficiently by parametric quadratic programming. However, there are many real-world nonlinear constraints which limit the number of different assets in a portfolio. Since the number of brands in the stock market is generally very large, techniques for selecting the effective portfolio are likely to be of interest in the financial field.

In recent decades, various approaches in the Artificial Intelligence (AI) field have been applied to several financial problems, especially for the stock market activities. Generally speaking, with the increasing need for more efficient portfolio selection and

stock trading models, AI approaches have been confirmed to outperform the conventional statistical models [68]-[71] in terms that they overcome the limitation of assumptions [85], [86].

Due to the bottlenecks of traditional methods, GNP [17], [87] and GRA [88] are proposed to solve these problems. Concretely speaking, the efficiency of GNP for generating stock trading rules has been confirmed in our previous studies [60], [89]. Also, GRA is originally developed to reduce a large class association rule set for data mining [88]. In this paper, we firstly apply GRA to the portfolio selection problem. In order to pick up the most efficient portfolio from a large number of brands, the proposed model considers the correlation coefficient between stock brands as strength, which indicates the relation between nodes in GRA. The algorithm evaluates the relationships between stock brands using a specific measure of strength and generates the optimal portfolio in the final generation. The performance of the portfolio selected with GRA is confirmed by the stock trading model with GNP. Generally speaking, the contributions of the proposed method are as follows.

- First, in the conventional Markowitz methods, a combination of brands is given, then the distribution ratio of the capital to each brand is determined by considering the return and risk. However, in our method, the best combination of stocks is selected from a large number of brands by considering the correlation coefficients of the brands in GRA. Then, the distribution of initial capital and the best strategy for buying and selling stocks are determined by GNP.

- Second, though many AI approaches have been applied to this field, most portfolio selection models in the previous literature only consider the distribution property of investment returns; other factors, such as investors' risk preferences and trading strategies, are not taken into account. Unlike others, these important factors are considered in our study. In a sense, the proposed model is an integrated intelligent model.

- Third, the number of stock brands in the best portfolio in the final generation of GRA can be flexibly defined by users since the brands correspond to nodes in the GRA individuals.

The further study is organized as follows. Section 5.2 describes the proposed GRA

approach in general. In Section 5.3, we explain the application of GRA to our proposed portfolio selection model. Section 5.4 describes the stock trading strategy of GNP, which is used to verify the GRA approach. Section 5.5 presents experimental environments, conditions and results using the integrated intelligent model. The trading profits are presented and compared with other methods. Finally, Section 5.6 concludes this chapter.

5.2 Genetic Relation Algorithm

In this section, the outline of GRA is explained in brief. Basically, GRA is an extension of GP [6] and GNP [17] in terms of gene structures. The original idea is based on the more general representation ability of both directed and undirected graphs. As a new evolutionary computation method, GRA is used to extract the events from a large number of candidates, which shows the best relations with each other in a GRA individual. When it is applied to the portfolio selection, GRA is used to select the optimal portfolio out of a huge number of possible stock brands. There are two kinds of gene structures in GRA, i.e., GRA with directed and undirected edges.

5.2.1 GRA with Directed Edges

Fig. 5.1 shows the basic structure and genotype expression of GRA with directed edges. GRA is composed of nodes and edges, where nodes represent events and directed edges represent the relations between nodes with their strength. As shown in Fig. 5.1, node i has strength S_{ij} to node j and node j has strength S_{ji} to node i.

Fig. 5.1 also describes the gene of node i, then the set of these genes represents the genotype of GRA individuals. Concretely speaking, ID_i represents an identification number of the node, e.g., ID_i=1 means node i has the directed edges to other nodes, while ID_i=2 means node i has the undirected edges to other nodes. F_i denotes the function of node i. In this chapter, F_i represents different stock brands in the portfolio. C_{i1}, C_{i2}, ..., C_{ik} show the number of the nodes which are connected from node i firstly, secondly and so on. S_{i1}, S_{i2}, ..., S_{ik} denote the strength of edges from node i to node C_{i1}, C_{i2}, ..., C_{ik}, respectively. All GRA individuals in a population have the same number of nodes.

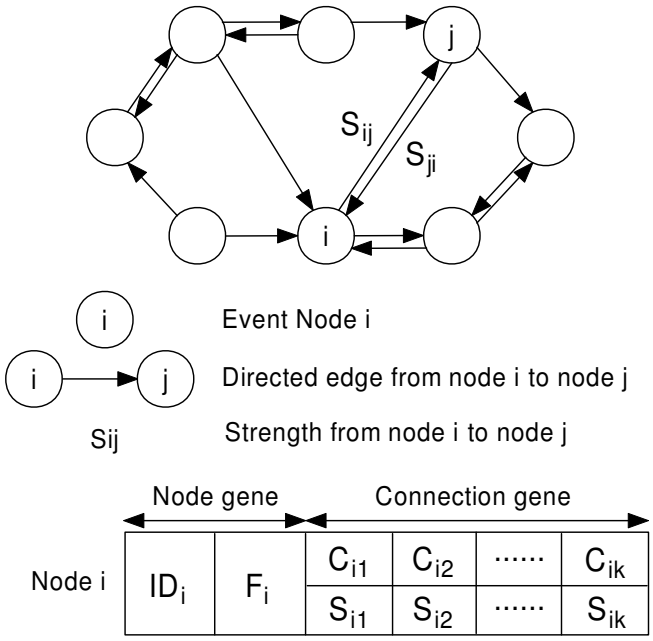


Figure 5.1: Basic structure of GRA with directed edges

Like other evolutionary algorithms, selection, crossover and mutation are used as the genetic operators of GRA. The outline of evolution is described as follows:

(1) Initialize the first population and calculate the fitness of the population;

(2) Generate new individuals for the next generation by tournament selection and genetic operations of crossover and mutation;

(3) Calculate the fitness of the new individuals;

(4) Repeat 2-3 until the terminal condition meets.

The points of GRA can be described as follows: First, GRA extracts appropriate events from a large number of candidates which have the best relations in a individual. Second, all the connections between nodes do not have to be defined, but the connection itself could be evolved. Also, the number of connections between nodes is flexible and it can be defined by users.

5.2.2 GRA with Undirected Edges

Fig. 5.2 shows the basic structure of GRA with undirected edges. Like the directed GRA, the event is also represented by the node, while the relation between nodes is

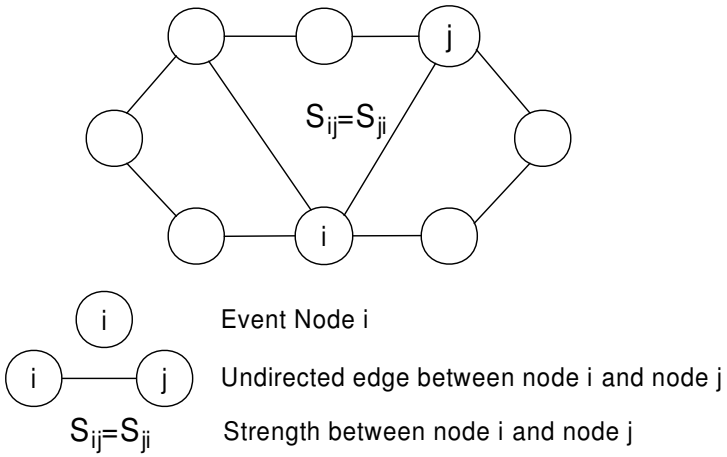


Figure 5.2: Basic structure of GRA with undirected edges

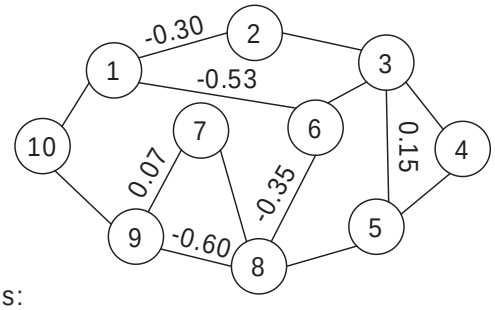
represented by undirected edges with their strength. The relation between node *i* and node *j* has a strength of S_{ij}=S_{ji} in undirected GRA, which is different from directed one.

5.3 Portfolio Selection using Genetic Relation Algorithm

In the proposed method, genetic relation algorithm with undirected edges are used to construct the portfolio selection model. As shown in Fig. 5.3, the basic structure of GRA is described as follows: The nodes in GRA are used to represent different stock brands in a portfolio, and the strength between two nodes are used to indicate the relationship between stock brands, i.e., the value of correlation coefficient. The main point of the proposed model is to select a given number of stocks in a portfolio from a large number of brands, so that the correlations among the stock brands in GRA satisfy a certain criterion. In order to maximize the final profit by the buying and selling strategy of GNP [60], we can study what degree of correlation coefficient the stocks should have by GRA method.

5.3.1 Notations and Fitness Function of GRA

- D: set of days
- S: set of stock brands
- $S(G)$: set of stock brands in GRA



Examples:

node1 -- node2: S_{12} = -0.30 node3 -- node5: S_{35} = 0.15

node7 -- node9: S_{79} = 0.07 node6 -- node8: S_{68} = -0.35

node1 -- node6: S_{16} = -0.53 node8 -- node9: S_{89} = -0.60

Node function: Stock brand

Strength: Correlation coefficient between stock brands

Figure 5.3: Genetic relation algorithm for portfolio selection

- $S(G_i)$: set of stock brands whose strength is defined between node i in GRA
- $Price(i,d)$: price of stock brand i on day d
- μ_i: mean of the price of stock brand i
- σ_i^2: variance of the price of stock brand i
- σ_{ij}: covariance between the prices of stock brand i and stock brand j
- ρ_{ij}: correlation coefficient between the prices of stock brand i and stock brand j
- ρ: target value of the correlation coefficient

The object of GRA is to select appropriate $|S(G)|$ stock brands out of a large number of brands $|S|$, which satisfy a certain value of the correlation coefficient, i.e., -1.0$\leq\rho\leq$1.0. Therefore, the fitness function of GRA is defined as follow.

$$Fitness = \frac{1}{|S(G)|}\sum_{i\in S(G)}\frac{1}{|S(G_i)|}\sum_{j\in S(G_i)}(\rho_{ij}-\rho)^2, \tag{5.1}$$

where,

$$\rho_{ij} = \frac{\sigma_{ij}}{\sigma_i\sigma_j},$$

$$\sigma_i^2 = E[(Price(i,d) - \mu_i)^2] = \frac{1}{|D|}\sum_{d\in D}(Price(i,d) - \mu_i)^2,$$

$$\sigma_{ij} = E[(Price(i,d) - \mu_i)(Price(j,d) - \mu_j)]$$
$$= \frac{1}{|D|}\sum_{d\in D}(Price(i,d) - \mu_i)(Price(j,d) - \mu_j),$$

$$\mu_i = E[Price(i,d)] = \frac{1}{|D|}\sum_{d\in D} Price(i,d).$$

In the fitness function of Eq. (1),

- if ρ is around 1.0, then stock brand i and stock brand j have positive correlation.
- if ρ is around -1.0, then stock brand i and stock brand j have negative correlation.
- if ρ is around 0.0, then stock brand i and stock brand j have no correlation.

The fitness function evaluates the GRA individuals so that the strengths between stock brands have the target value of the correlation coefficient ρ. Generally, according to the portfolio theory, it is preferable to select $|S(G)|$ stock brands which have small correlations. It is our interest to find out the target value of the correlation coefficient ρ in the fitness function. By the portfolio selection model of GRA, the stock brands having large correlations with each other are expected to be eliminated, as they always cause high risk in a portfolio.

5.3.2 Genetic Operators of GRA

In this sub-section, the genetic operators in the evolution phase are introduced. In order to get the best individual, the function of nodes in GRA should be changed, which can be realized effectively by genetic operations. GRA has three kinds of genetic operators: selection, crossover and mutation. In GRA, mutation operation could be executed not only on the connections between nodes but also on the node functions.

Selection

At each generation, all of the individuals are ranked by their fitness values and the best individual in the current generation is preserved for the next generation by elite selection. Then, tournament selection of individuals is carried out to reproduce the next generation.

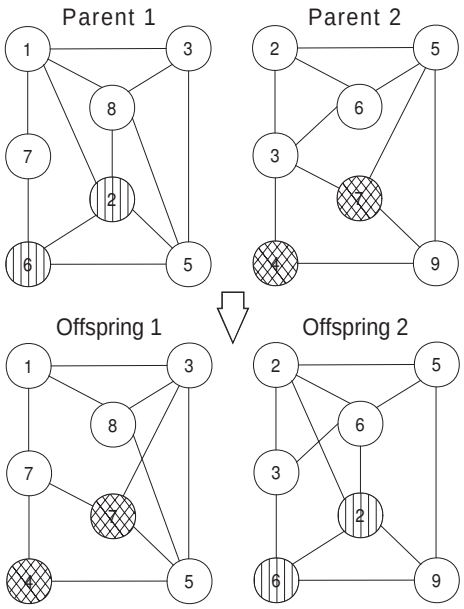


Figure 5.4: Crossover

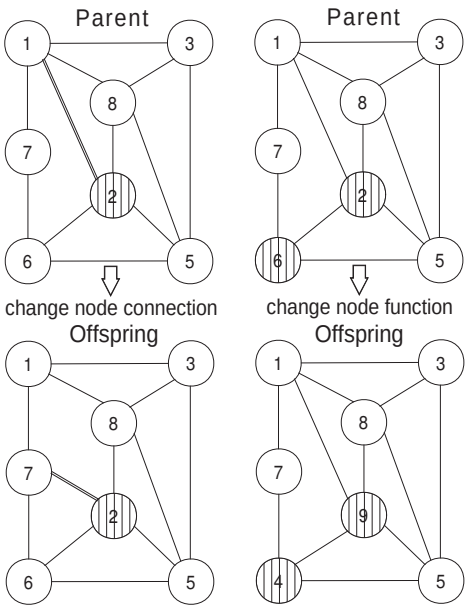


Figure 5.5: Mutation

Crossover

As shown in Fig. 5.4, crossover is executed between two parents and two offspring are generated. The procedure of crossover is as follows.

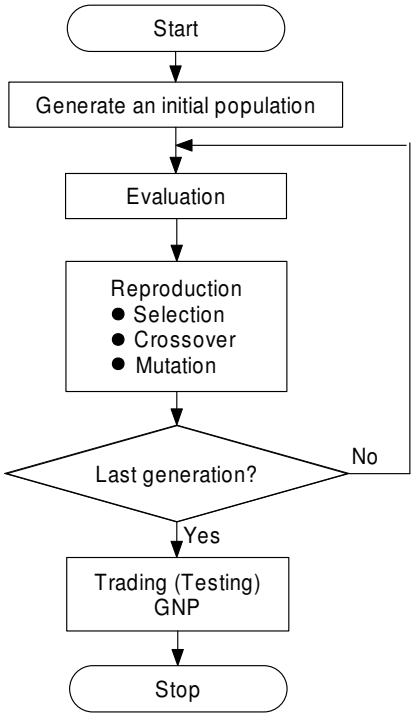


Figure 5.6: Flowchart of GRA

- Select two individuals using tournament selection twice and produce them as parents.
- Each node is selected as a crossover node with the probability of P_c.
- Two parents exchange the genes of the corresponding crossover nodes.
- Generated new individuals become the new ones of the next generation.

Mutation

Fig. 5.5 shows an example of the mutation operator. Mutation is executed in one individual and a new one is generated. The procedure of mutation is as follows.

- Select one individual as a parent using tournament selection.
- Mutation operation
  - change connection: Each node branch (C_{i1}, C_{i2}, ..., C_{ik}) is selected with the probability of P_m, and the selected branch is reconnected to another node.

  - change node function: Each node function (F_i) is selected with the probability of P_m, and the selected function is changed to another one.

- Generated new individual becomes the new one of the next generation.

5.3.3 Flowchart of GRA

Fig. 5.6 shows the flowchart of GRA. For the first GRA population, each individual is generated assigning a certain stock brand selected randomly from a huge number of brands to one of the nodes in GRA. It is ensured that all nodes are different within one individual. In the next, evaluation of the individuals is carried out according to their fitness values. At the reproduction phase, selection, crossover and mutation are used as genetic operators to generate the population for the next generation. This process is repeated until the last generation. Finally, after obtaining the best individual in the last generation, it is tested by the stock trading model of GNP [60].

5.4 Stock Trading Strategy of Genetic Network Programming with Control Nodes

In this section, a stock trading strategy based on GNPcn [17], [60], [90] is used for evaluating the proposed GRA portfolio selection method. It is a kind of integrated intelligent model by considering both portfolio selection and the trading strategies. The overall procedure for the GNPcn approach is presented as follows.

- Firstly, the role of GNPcn is to determine the distribution of the initial capital to each brand selected by GRA, and also determine the time of buying and selling stocks.

- Secondly, since GNPcn is evolved with training data, the brands which obtain larger profitability in the training period can have the initial capital more than other brands in the portfolio. As a result, the distribution ratio of the capital and the time of buying and selling stocks are determined by the transition of judgment nodes and processing nodes in GNPcn.

- Thirdly, as the features of GNPcn method, the combination of evolution and learning is realized to take the appropriate trading actions, i.e., reinforcement

Table 5.1: Parameter conditions for evolving GRA

Number of individuals=300 (mutation:179, crossover:120, elite:1)
Number of generations=300
Number of nodes=10
P_c=0.3, P_m=0.1

learning is used to select subnode functions in GNPcn according to their Q values. Moreover, technical indices and candlestick chart are used as judgment functions to get the information from the stock market depending on the situation, thus, the stock trading strategy of GNPcn can be carried out effectively.

Since the stock trading strategy of GNPcn was described in detail in Chapter 4 and paper [90], we use the same algorithm and simulation settings in this chapter and will not repeat the procedure again.

5.5 Experimental Results

In order to confirm the effectiveness of GRA for the portfolio selection model, we carried out the trading simulations by GNP using the best GRA individual that was obtained in the last generation. The simulation is divided into two steps: one is used for the training of GRA and the other is used for the training and testing of GNP.

- Training (GRA): January 4, 2001—December 30, 2003 (737 days)
- Training (GNP): January 4, 2001—December 30, 2003 (737 days)
- Testing (GNP): January 5, 2004—December 30, 2004 (246 days)

5.5.1 Performance of Genetic Relation Algorithm

Experimental Conditions of GRA

Table 5.1 shows the parameters of the evolution of GRA. The total number of nodes in each individual of GRA is 10 which indicate 10 different stock brands in a portfolio. Those stock brands are selected from 500 companies listed in the first section of

Table 5.2: Profit and profitability comparison using different number of branches (yen)

ρ value	1 branch	full branches
-0.8	2,766,084 (5.53%)	2,631,280 (5.26%)
-0.6	2,672,986 (5.35%)	2,897,513 (5.80%)
-0.4	3,179,176 (6.36%)	3,300,970 (6.60%)
-0.2	3,384,061 (6.77%)	3,261,387 (6.52%)
0.0	3,797,128 (7.59%)	3,673,806 (7.35%)
0.2	3,502,800 (7.01%)	3,607,220 (7.21%)
0.4	2,973,384 (5.95%)	3,180,279 (6.36%)
0.6	3,084,725 (6.17%)	2,862,180 (5.72%)
0.8	2,580,135 (5.16%)	2,500,410 (5.00%)

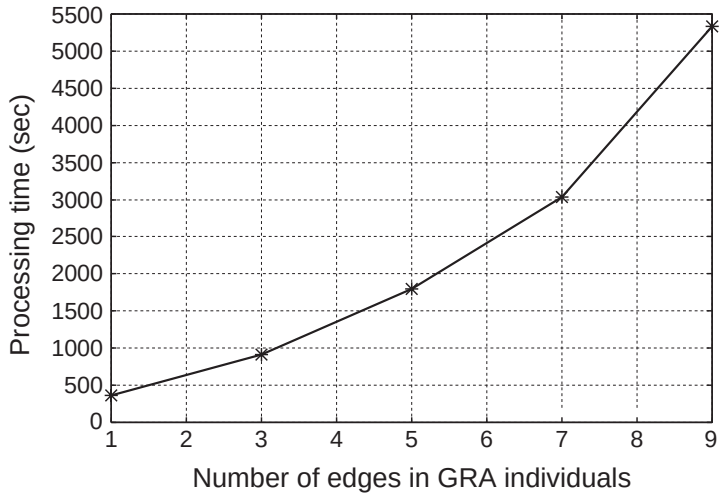


Figure 5.7: Processing time when changing the number of edges in GRA

Tokyo stock market in Japan. The content F_i in each node, i.e., the stock brand, is determined randomly at the beginning of the first generation, and changed appropriately by evolution.

The initial connections between nodes are also determined randomly at the first generation. At the end of each generation, 179 new individuals are produced by mutation, 120 new individuals are produced by crossover, and the best individual is preserved. The other parameters for crossover and mutation are the ones showing good results in the simulations. The terminal condition is 300 generations.

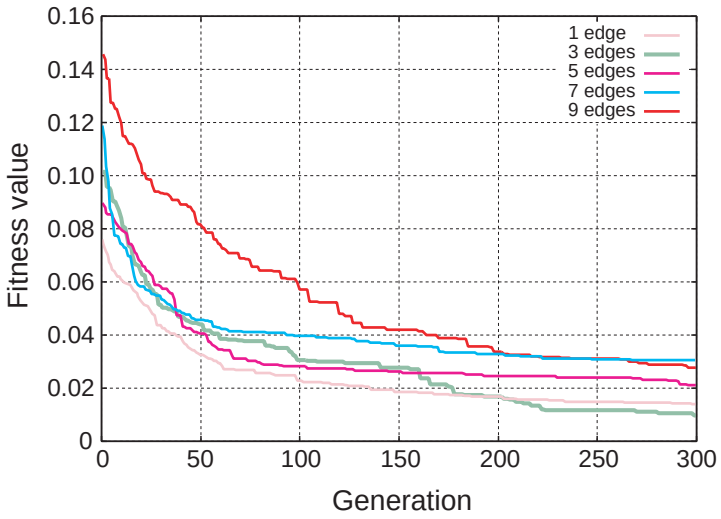


Figure 5.8: Average fitness value when changing the number of edges in GRA

Simulation Results of GRA

Fig. 5.7 shows the average processing time when the number of edges in a GRA individual is changed. It is an example when the target correlation coefficient æ is set to 0.0. It is clear from Fig. 5.7 that when the number of edges increases, the average processing time also increases because of the complexity of the network structures.

Fig. 5.8 shows the average fitness values when the number of edges in a GRA individual is changed using the data from 2001 to 2003, and these lines are the average values over 30 independent simulations. From Fig. 5.8, we can see that the differences of the fitness values between one edge and the large number of edges become small as the generation goes on.

Table 5.2 shows the profitability comparison using different number of edges with different value of correlation coefficient æ, while the initial budgets for training and validation, i.e., $Initial(t)$ and $Initial(v)$ are equal to 50,000,000 Japanese yen. The profitabilities are the simulation results obtained with the integrated intelligent model consisting of GRA and GNP. From Table 5.2, it is found that comparable results are obtained with only one edge compared to full edges of each node in GRA individual. Since a small number of edges can save the processing time as shown in Fig. 5.7, and comparable fitness value and profitability are obtained in general, it is unnecessary to consider the connections of full edges between nodes. Therefore, only one branch is used for each

Table 5.3: Parameter conditions for evolving GNP

Number of individuals=300 (mutation=179, crossover=120, elite=1)
Number of nodes=80 (judgement node=20, processing node=10, control node=50)
Number of sub-node in each node=2
P_c=0.1, P_m=0.03, α=0.1, γ=0.3, ϵ=0.1

node in the evolution of GRA, which can be evolved by the crossover and mutation.

5.5.2 Validation By the Stock Trading Model of Genetic Network Programming

Experimental Conditions of GNP

Table 5.3 shows the parameters for the evolution of GNP method. GNP uses the judgment nodes to judge the information from stock markets, and uses the processing node to take buying and selling actions. Five control nodes are assigned to each brand. The total number of nodes in each individual is 80 including 10 processing nodes, 20 judgment nodes and 50 control nodes. The initial connections between nodes are also determined randomly at the first generation. At the end of each generation, new individuals are produced by selection, crossover and mutation. The initial Q values are set at zero.

In the validation phase of the stock trading model of GNP, we suppose that the initial funds are 50,000,000 Japanese yen in both training and testing periods. Especially, when we use GNP to test the best portfolio generated by GRA, one GNP individual has 10 groups of control nodes, each of which deals with one brand, so one GNP can deal with 10 brands in the portfolio simultaneously.

Simulation Results of GNP

In order to confirm the efficiency of our proposed method, Table 5.4 shows the comparison of the profit between GNP with GRA and conventional GNP. Concretely speaking, in the case of GNP with GRA, we carry out the simulations by setting the various values of ρ in the fitness function Eq. (1). The value of ρ indicates positive or negative correlation between different stock brands, and it has strong effects to the portfolio selection. Concretely speaking, Table 5.4 presents the average profit of the portfolio selected by

Table 5.4: Comparison of profit with conventional GNP (Profit[yen])

ρ	-0.02	-0.04	-0.06	-0.1	-0.2	-0.4	-0.6	-0.8
GNP with GRA	3,501,807	3,251,973	3,163,410	3,352,168	3,384,061	3,179,176	2,672,986	2,766,084
ρ	0.0							
GNP with GRA	3,797,128							
ρ	0.02	0.04	0.06	0.1	0.2	0.4	0.6	0.8
GNP with GRA	3,451,030	3,600,963	3,180,345	3,275,730	3,102,800	2,973,384	3,084,725	2,580,135
random sequence	1	2	3	4	5	6	7	8
conventional GNP	1,738,324	2,631,634	1,062,671	2,987,880	3,148,120	900,380	1,220,368	2,733,425
random sequence	9	10	11	12	13	14	15	16
conventional GNP	1,376,708	3,337,781	751,993	2,602,448	1,573,080	1,992,164	3,029,950	1,259,492

Table 5.5: Stock brands in the optimal portfolio selected by GRA

a	Nissin Foods Products Co., Ltd.
b	Hitachi Chemical Co., Ltd.
c	ToTo Ltd.
d	Toshiba Corporation
e	Honda Motor Co., Ltd.
f	Yamaha Corporation
g	Toyota Tsusho Corporation
h	All Nippon Airways Co., Ltd.
i	Chubu Electric Power Co., Inc.
j	Toho Co., Ltd.

GRA over 30 independent simulations when the correlation coefficient ρ is set at different values. From the results, it is clarified that we can get a good profit in the testing period when ρ is set at 0.0. Therefore, we set the value of 0.0 for the parameter ρ in our simulations. In the case of conventional GNP as shown in Table 5.4, we randomly select 16 portfolios without the optimization model of GRA from 500 companies listed in the first section of Tokyo stock market in Japan. From Table 5.4, the portfolio selected by the GRA optimization model with small $|\rho|$ can obtain higher profits than conventional GNP, which selected stock brands randomly from the stock market. Moreover, in the previous study, it has been confirmed that GNP outperformed the other traditional methods [60], [40] and benchmark, i.e., GA and Buy&Hold, which are widely used in the financial field. Thus, we didn't carry out the comparisons between GNP and other traditional methods in this chapter.

Fig. 5.9 shows the fitness and profit curves, i.e., the best $Fitness(n)$ and $Profit(t, b, n)$

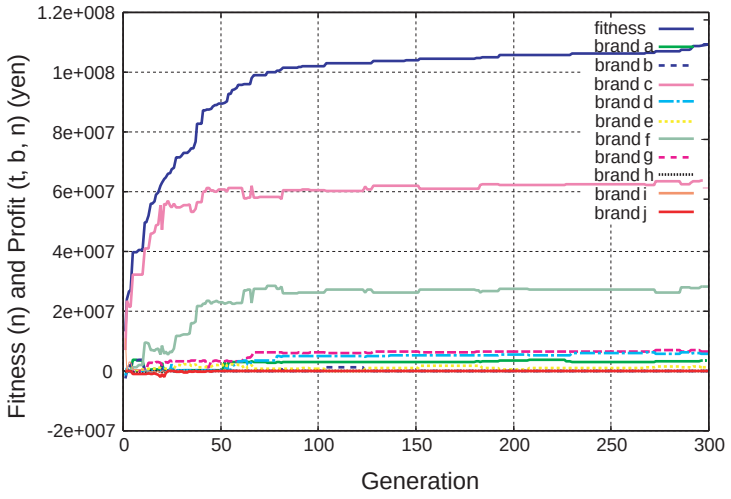


Figure 5.9: Fitness and profit curves of 10 brands in the training period by GNP

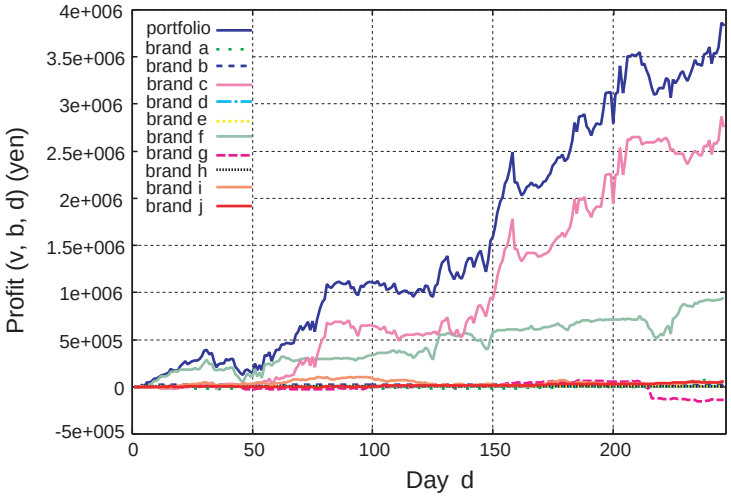


Figure 5.10: Profits change of selected 10 brands in the testing period by GNP

of each brand $b \in S(G)$ in the *nth* generation using the data from 2001 to 2003, and the line is the average value over 30 independent simulations. From Fig. 5.9, we can see that GNP with reinforcement learning can obtain larger profits for the training data as the generation goes on.

Fig. 5.10 shows the profits change $Profit(v, b, d)$ of the selected 10 brands in the testing period by GNP model, i.e., using the best portfolio obtained with GRA method when the value of parameter ρ is set to 0.0. We carried out the dealing of these 10 brands using the data of 2004. From Fig. 5.10, we can see that the profit keep increasing during

the testing period. The advantage of the proposed integrated intelligent model is that it can optimize not only the portfolio selection, but also optimize the distribution of initial budget to each brand in the portfolio automatically, and the brands which obtain larger profitability can have the initial budget more than other brands. As a result, with this efficient portfolio optimization system, we can obtain much profits in the trading of those brands. The stock brands in the optimal portfolio selected by GRA method are also shown in Table 5.5.

5.6 Summary

In this chapter, we applied Genetic Relation Algorithm and Genetic Network Programming to the portfolio selection problem. In order to pick up a fixed number of the most efficient brands, the algorithm evaluates the relationships between stock brands using a specific measure of strength, i.e., correlation coefficient between stocks, and generate the optimal portfolio in the final generation. We carried out the experiments selecting 10 brands out of 500 companies listed in the first section of Tokyo stock market in Japan for 4 years. In the experiments, the efficiency of GRA method is confirmed by the stock trading model of GNP that has been proposed in the previous chapter. Compared to conventional methods, the proposed integrated intelligent model has two advantages, first, GRA method considers the correlation coefficient as the strength between stock brands to optimize the portfolio, which is different from the conventional method that select stocks randomly for trading. From the results, it is clarified that we can obtain much profits in the trading of those brands. Second, GNP can adjust the distribution of the initial budget to each selected brand in the portfolio in each generation, and more budget is assigned to the brands with larger profitability. It is different from the Markowitz method in terms that the distribution ratio of the capital to each brand is determined for maximizing the total profits since the risk factor is already considered in GRA. Thus, we can create an effective portfolio optimization system and get more profits by the combination of GRA and GNP.

There remain some further studies in the future. They include, but not restricted to:

- First, the algorithm presented can be further improved by incorporating several new methods to allow the evolution of better GRA individuals, such as modify-

ing the genetic operators of crossover and mutation.

- Second, the portfolio selection model can be enhanced by modifying the fitness function, and it is necessary to study various measures in addition to the correlation coefficient.

- Third, the system will be tested on many kinds of markets and with different data periods.

Moreover, the system will incorporate additional financial analysis tools to increase the information content from the market.

Chapter 6

Genetic Relation Algorithm with Guided Mutation for the Large Scale Portfolio Optimization

6.1 Introduction

Portfolio optimization consists of the portfolio selection problem in which we want to find the optimum way of investing a particular amount of money in a given set of securities or assets [85]. Markowitz [64], [66] formulated the fundamental framework of a mean-variance portfolio framework, which explains the trade-off between mean and variance, representing expected returns and risk of a portfolio, respectively. The standard mean-variance model is based on the assumption that investors are risk averse and the return of assets are normally distributed. Although the task of yielding the minimum risk and maximum return looks simple, there is more than one way of establishing an optimum portfolio. Konno and Yamazaki [91] introduced an advanced model in which a mean-absolute deviation (MAD) model is utilized as a measure of risk. However, it was insensitive to some extremes, which could be the source of serious errors, contrary to the suggestion that the MAD model is suitable under all circumstances [92]. Since the stock market is a highly nonlinear dynamic system, it is affected by many factors such as interest rates, inflation rates, economic environment, and many others. Although there are dependencies and correlations between these factors, the optimum portfolio is rather difficult to construct through mathematic formula only, thus, techniques for selecting the effective portfolio have been developed in the recent years.

During the last few decades, stock traders have largely been relying on various types of intelligent systems to make trading decisions. Recently, many soft computing (SC) methods have been applied to the portfolio selection and stock trading systems. Atiya [93] discussed assessing credit risk and prediction using Artificial Neural Network (ANN), and Lam [74] applied the back-propagation algorithm to integrate the fundamental and technical analysis for financial performance prediction. The experimental results showed that the ANN outperforms the benchmark. Baba et al. [94] employed Neural Network (NN) and Genetic Algorithm (GA) to construct an intelligent decision support system (DSS) for analyzing the Tokyo Stock Price Index (TOPIX). The essential feature of their DSS is that it projects the high and low TOPIX values into the future and suggests buy and sell decisions based on the average projected value and the current value of the TOPIX. An intelligent stock trading decision support system that can forecast the buying and selling signals according to the prediction of short-term and long-term trends using rule-based NNs was developed by Chou et al. [95]. However, limitations of these models are due to the buried noise and complex dimensionality of stock price data. That is to say that the quantity of data itself and the input variables interfere with each other. Therefore, results were usually not as convincing. Moreover, Kimoto et al. [96] presented an improved NN and fuzzy models used for exchange rate prediction. A Takagi-Sugeno-Kang (TSK)-type fuzzy-rule-based system was developed by Chang and Liu [97] for stock price prediction. The TSK fuzzy model used the technical index as the input variables and the consequence is a linear combination of the input variables. In the paper [98], fuzzy logic and ANN were integrated into the fuzzy back-propagation network (FBPN) for sales forecasting in Printed Circuit Board (PCB) industry. However, the limitation of fuzzy logic is that, when designing the fuzzy models for stock trading, we need to get the expert knowledge prior to it.

Though many Artificial Intelligence (AI) approaches have been applied to this field, most portfolio selection models in the previous literatures only consider the distribution property of investment returns; other factors, such as investors' risk preferences and trading strategies, are not taken into account. However, the trading decision plays a critical role in the stock market environment if an investor wants to make a profit. Up to now, there are only a few examples of research that deal with the trading strategy of a stock, and notably, this problem is still a large one for academic researchers and

industrial practitioners. Therefore, this research takes a different approach by applying Genetic Relation Algorithm with guided mutation (GRA/G) and Genetic Network Programming (GNP) to construct a portfolio selection and stock trading model. In a sense, the proposed model is an integrated intelligent model.

In this chapter, we will discuss the portfolio optimization problem based on GRA and propose a new operator called guided mutation. Guided mutation can be regarded as an improvement of the conventional mutation operators, with which the resultant solution can hopefully fall in or close to a promising area. In such a way, the similarity between an offspring and its parent can be controlled to some extent. The main features of GRA/G are as follows.

(1) Genetic relation algorithm: GRA considers the correlation coefficients between stock brands as strength, which indicates the relation between nodes in each individual of GRA. The algorithm evaluates the relationships between stock brands using a specific measure of strength and generates the optimal portfolio in the final generation. The efficiency of GRA/G is validated by the stock trading model using Genetic Network Programming (GNP) which has been proposed in our previous study [90].

(2) Guided mutation: Guided mutation generates offspring according to the average value of correlation coefficients in each individual, which means to enhance the exploitation ability of evolution of GRA. New solutions are generated by applying the guided mutation operator to the current population, and the resultant solutions will not be far from the best portfolio found so far and fall into a promising area.

(3) Portfolio size: A remarkable feature of the algorithm is that it is extremely easy to implement, and it can be extended to any size of portfolio selection problems for finding the optimal solution since the number of stock brands in the best portfolio can be flexibly defined by users.

The rest of this chapter is organized as follows. Section 6.2 describes the proposed GRA/G approach in general. In Section 6.3, the application of GRA/G to the portfolio optimization is presented. In Section 6.4, the GNP stock trading model is briefly described. Section 6.5 examines the experimental tests conducted to determine the best

portfolio and its trading strategy. Lastly, conclusions and future work are provided in Section 6.6.

6.2 Genetic Relation Algorithm with Guided Mutation

The outline of genetic relation algorithm with guided mutation (GRA/G) is explained briefly in this section. Basically, GRA/G is an extension of Genetic Programming [6] and Genetic Network Programming (GNP) [17] in terms of gene structures. The original idea is based on the more general representation ability of both directed and undirected graphs. As a new evolutionary computation, generally speaking, GRA/G is used for determining the best relations between events. Therefore, GRA/G is used to extract a fairly small number of events from a large set of events, where an event is represented and encoded by nodes in GRA/G. In the real world applications, the events are defined depending on the problem to solve, for instance, in data mining [88], the nodes represent association rules, and in the stock market, the nodes represent stock brands in a portfolio. The main limitation of GRA/G is the complexity in its structure when the number of nodes and edges increases. In order to overcome this problem, we can make the number of edges for each node as small as possible to guarantee the performance. In this chapter, our target is to pick up the most efficient portfolio from a large number of stock brands by considering the correlation coefficient between stock brands as strength, i.e., the relation between nodes in GRA/G. As a new genetic operator, guided mutation is used to generate offspring according to the average value of strength in each individual to enhance the exploitation ability of evolution. As explained in the following, there are two kinds of gene structures in GRA/G, i.e., GRA/G with directed and undirected edges.

6.2.1 GRA/G with Directed Edges

Fig. 6.1 shows the basic structure and genotype expression of GRA/G with directed edges. GRA/G is composed of nodes and edges, where nodes represent events and directed edges represent the relations between nodes with their strength. As shown in Fig. 6.1, node i has strength S_{ij} to node j and node j has strength S_{ji} to node i.

Fig. 6.1 also describes the gene of node i, then the set of these genes represents the genotype of GRA/G individuals. Concretely speaking, ID_i represents an identification

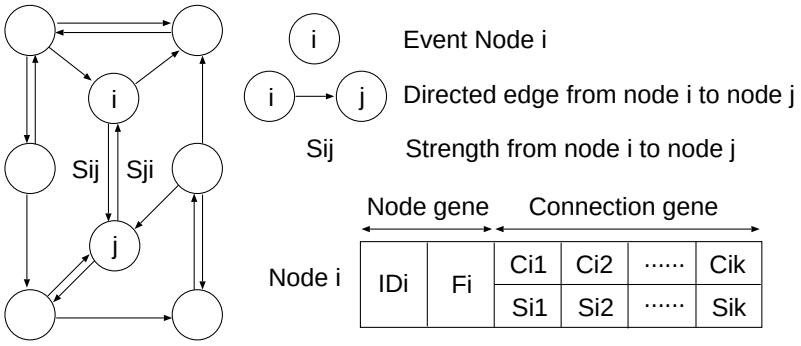


Figure 6.1: Basic structure of GRA/G with directed edges

number of the node, e.g., ID_i=1 means node i has the directed edges to other nodes, while ID_i=2 means node i has the undirected edges to other nodes. F_i denotes the function of node i. In this chapter, F_i represents different stock brands in the portfolio. C_{i1}, C_{i2}, ..., C_{ik} show the nodes which are connected from node i firstly, secondly and so on. S_{i1}, S_{i2}, ..., S_{ik} denote the strength of edges from node i to node C_{i1}, C_{i2}, ..., C_{ik}, respectively. All individuals in a population have the same number of nodes. Traditional GRA with directed edges has been applied to the data mining by Gonzales [88], where GRA is used for association rules with non-fixed consequent. In that case, the antecedent and consequent of each rule are represented as nodes in GRA, while the confidence of each rule could be used as the strength.

Like other evolutionary algorithms, selection, crossover and mutation are used as the genetic operators of GRA/G. The outline of evolution is described as follows:

(1) Initialize a randomly generated population.

(2) Evaluate the fitness of individuals in the population.

(3) Generate new individuals for the next generation by tournament selection and genetic operations of crossover and guided mutation.

(4) Replace the current population by the new population.

(5) If the termination condition is satisfied then stop, else go to step 2.

One important point of GRA/G is that all the connections between nodes do not have to be defined, but the connection itself could be evolved.

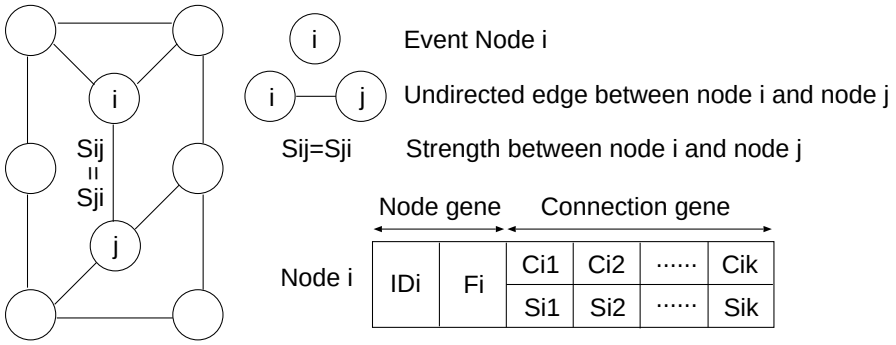


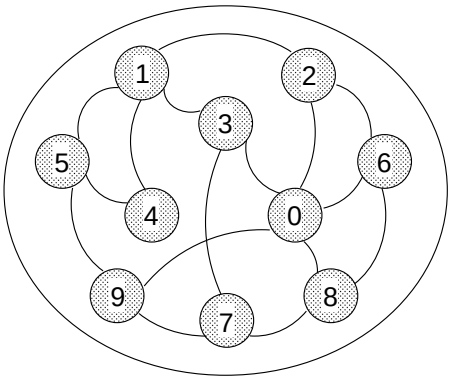
Figure 6.2: Basic structure of GRA/G with undirected edges

6.2.2 GRA/G with Undirected Edges

Fig. 6.2 shows the basic structure of GRA/G with undirected edges. Same as directed GRA/G, the event is also represented by the node, while the relations between nodes are represented by undirected edges with their strength. The relation between node i and node j has a strength of S_{ij}=S_{ji} in GRA/G with undirected edges, which is different from directed GRA/G. As shown in Fig. 6.2, the set of genes represents the genotype of GRA/G individuals with undirected edge. ID_i represents an identification number of the node and F_i denotes the function of node i. C_{i1}, C_{i2}, ..., C_{ik} show the nodes which are connected from node i firstly, secondly and so on. S_{i1}, S_{i2}, ..., S_{ik} denote the strength of edges from node i to node C_{i1}, C_{i2}, ..., C_{ik}, respectively. In this chapter, GRA/G with undirected edges is used for portfolio selection due to its special feature. That is, the relation between node i and node j has the same strength of S_{ij}=S_{ji}, while also in the stock market, the correlation coefficient between stock i and stock j has the same value.

6.3 Portfolio Selection using GRA/G

In the proposed method, GRA/G with undirected edges are used to construct the portfolio selection model. As shown in Fig. 6.3, the basic structure of GRA/G is described as follows: The nodes in GRA/G are used to represent different stock brands in a portfolio, and the strength between two nodes is used to indicate the relationship between stock brands, i.e., the value of correlation coefficient. The main point of the proposed model is to select a given number of appropriate stocks in a portfolio by making the absolute values of correlation coefficients of stock brands in GRA/G as small as possi-



S13=-0.31 S26=0.67 S68=-0.35

S95=-0.29 S78=0.56 S37=0.29

Node function: Stock brand

Strength: Correlation coefficient between stock brands

Figure 6.3: GRA/G for portfolio selection

ble, which leads to avoid the risk. In order to maximize the final profit of the portfolio, we use the buying and selling strategy of GNP [89], [90] for the stock trading.

6.3.1 Fitness Function of GRA/G

The object of GRA/G is to select appropriate $|S(G)|$ stocks out of a total number of stocks $|S|$. Therefore, the fitness function of GRA/G is defined as follow.

$$Fitness = \frac{1}{|S(G)|} \sum_{i \in S(G)} \frac{1}{|S(G_i)|} \sum_{j \in S(G_i)} \rho_{ij}^2, \tag{6.1}$$

where,

$$\rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j},$$

$$\sigma_i^2 = E[(Price(i,d) - \mu_i)^2] = \frac{1}{|D|} \sum_{d \in D} (Price(i,d) - \mu_i)^2,$$

$$\sigma_{ij} = E[(Price(i,d) - \mu_i)(Price(j,d) - \mu_j)] = \frac{1}{|D|} \sum_{d \in D} (Price(i,d) - \mu_i)(Price(j,d) - \mu_j),$$

$$\mu_i = E[Price(i,d)] = \frac{1}{|D|} \sum_{d \in D} Price(i,d).$$

In the above equations, D is the set of days, S is the set of stocks, $S(G)$ is the set of stocks in GRA/G, $S(G_i)$ is the set of stocks whose strength is defined between node i in GRA/G, $Price(i, d)$ is the price of stock i on day d, μ_i is the mean value of the price of stock i, σ_i^2 is the variance of the price of stock i, σ_{ij} is the covariance between the prices of stock i and stock j, and ρ_{ij} is the correlation coefficient between the prices of stock i and stock j. Especially, in the fitness function of Eq. (1),

- if ρ_{ij} is around 1.0, then stock i and stock j have positive correlation.
- if ρ_{ij} is around -1.0, then stock i and stock j have negative correlation.
- if ρ_{ij} is around 0.0, then stock i and stock j have no correlation.

The fitness function evaluates the GRA/G individuals if the strengths between stock brands have a small value of correlation coefficient ρ_{ij}. Generally, according to the portfolio theory, it is preferable to select $|S(G)|$ stocks which have small correlations. It is our interest to find out the stock brands with small absolute values of the correlation coefficient ρ_{ij} in the fitness function. By the portfolio selection model of GRA/G, the stocks having large correlations with each other will be eliminated, as they always cause high risk in a portfolio.

6.3.2 Genetic Operators of GRA/G

In this sub-section, the genetic operators in the evolution phase are introduced. In order to get the best individual, the function of nodes in GRA/G should be changed, which can be realized effectively by genetic operations. GRA/G has three kinds of genetic operators: selection, crossover and mutation. In GRA/G, mutation operation could be executed not only on the connections between nodes but also on the node functions.

Selection

At each generation, all of the individuals are ranked by their fitness values and the best individual at the current generation is preserved for the next generation by elite selection. Then, tournament selection of individuals is carried out for reproducing the next generation.

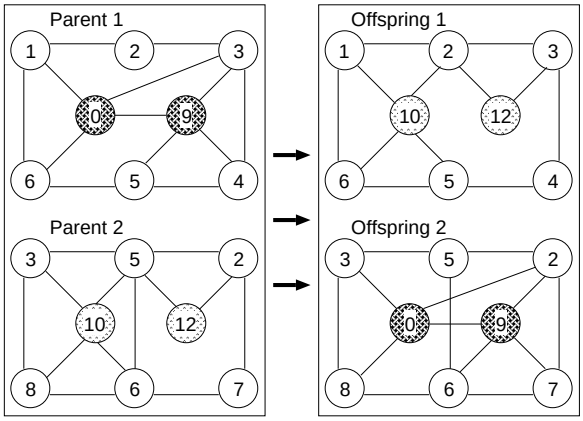


Figure 6.4: Crossover

Crossover

As shown in Fig. 6.4, crossover is executed between two parents and two offspring are generated. The procedure of crossover is as follows.

- Select two individuals using tournament selection twice and produce them as parents.
- Each node is selected as a crossover node with the probability of P_c.
- Two parents exchange the genes of the corresponding crossover nodes.
- Generated new individuals become the new ones of the next generation.

Guided Mutation

One of the key points in the design of GRA/G model is how to generate offspring in an effective way. The proximate optimality principle [99], an underlying assumption in most evolutionary methods, assumes that good solutions have the similar structure. This assumption is reasonable for most real-world problems. Based on this assumption, an ideal offspring generator should be able to produce a solution which is close to the best solutions. In the case of GRA/G, the idea behind the proposed operator which we call guided mutation is to generate offspring according to the average value of the correlation coefficients in each individual. New solutions (portfolios) are generated by applying the guided mutation operator to the current population, then the resultant

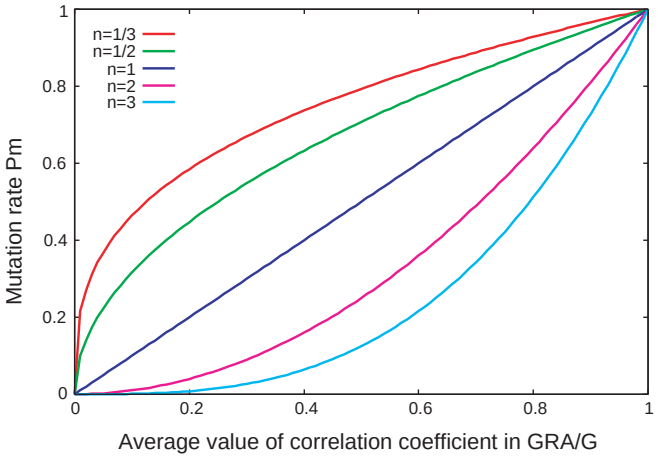


Figure 6.5: Guided mutation rate with different values of n

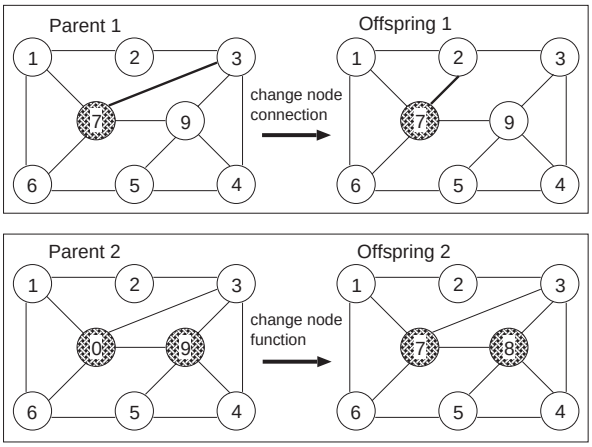


Figure 6.6: Mutation

solutions will be not far from the best portfolio and fall into a promising area. The mutation rate P_m for different GRA/G individual is calculated as follows.

$$P_m = \rho^n, (n = ..., 1/3, 1/2, 1, 2, 3, ...) \tag{6.2}$$

where,

$$\rho = \frac{1}{|S(G)|} \sum_{i \in S(G)} \rho_i,$$

$$\rho_i = \frac{1}{|S(G_i)|} \sum_{j \in S(G_i)} |\rho_{ij}|.$$

ρ is the average value of the correlation coefficients in GRA/G, while ρ_i is the average value of the correlation coefficients between the prices of stock i and other stocks in GRA/G. In the proposed method, guided mutation plays an important role of introducing diversity to the population, thus, the mutation rate should be appropriately adjusted in order to enhance the exploitation of the search to find better solutions.

Fig. 6.5 shows the guided mutation rate when changing the value of n in Eq. (2). Since the value of n may strongly affect the relation between correlation coefficients ρ and guided mutation rate P_m, an appropriate n value should be selected so as to get the optimal portfolios. Various values of n are tested in the later part of simulations.

Fig. 6.6 shows an example of the mutation operator. Mutation is executed in one individual and a new one is generated. The procedure of mutation is as follows.

- Select one individual as a parent using tournament selection.
- Mutation operation
  - change connection: Each node edge (C_{i1}, C_{i2}, ..., C_{ik}) is selected with the probability of P_m, and the selected edge is reconnected to the other node.
  - change node function: Each node function (F_i) is selected with the probability of P_m, and the selected function is changed to the other one.
- Generated new individual becomes the new one of the next generation.

6.3.3 Flowchart and Process of GRA/G

Fig. 6.7 shows the flowchart of GRA/G. For the first GRA/G population, each individual is generated by assigning certain stock brands selected randomly to each node of GRA/G. It is ensured that all nodes are different within one individual. In the next, evaluation of the individuals is carried out according to their fitness values. At the reproduction phase, selection, crossover and mutation are used as genetic operators to generate the population for the next generation. This process is repeated until the last generation. Finally, after obtaining the individuals in the last generation, they are tested by the stock trading model of GNP [89], [90].

In order to compare the difference between experts' decisions and the GRA/G approach, the detailed processes of trading decision's generation by financial experts and

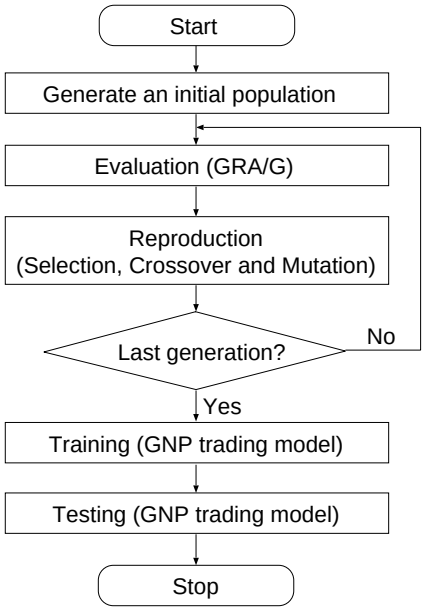


Figure 6.7: Flowchart of GRA/G

GRA/G are shown in Fig. 6.8. In traditional applications, the judgment of a trading signal was decided by financial experts' opinions. Thus, the financial experts base their trading decisions on the technical analysis. Thereby, if the trading signal is not judged properly, the investor can make a mistake and lose money. However, in our research, the portfolio selection and stock trading decisions can be directly generated from the intelligent model of GRA/G and GNP, with the expectation to make more profits. As explained in Fig. 6.8, the step-by-step procedure of the proposed approach can be described in the following:

- Step 1: Evaluate the relationships between stock brands using a measure of correlation coefficients and generate the portfolios in the final generation of GRA/G.
- Step 2: Test each portfolio by using the stock trading model of GNP [90], which considers the technical indices and candlestick chart as trading signals to make decisions.
- Step 3: Calculate the final profit of each portfolio, and choose the one with the largest profit as an optimal portfolio, i.e., the best combination of stock brands for the investors.

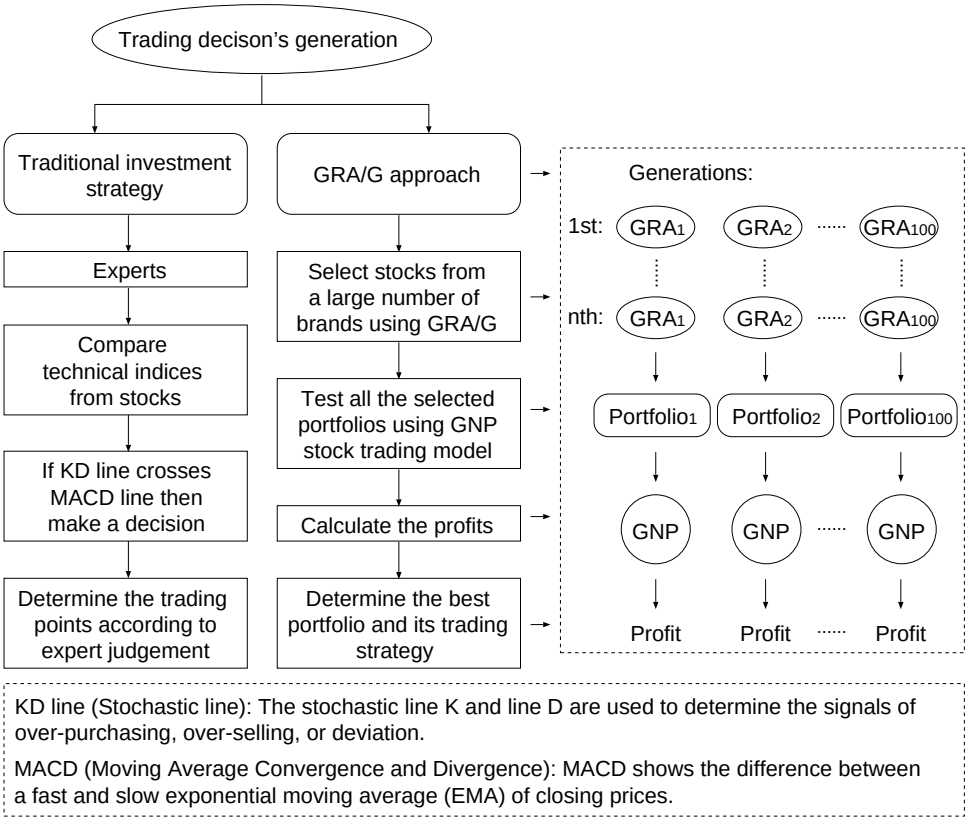


Figure 6.8: Processes of trading decision's generation by GRA/G and financial experts

6.4 Stock Trading Strategy of Genetic Network Programming

In this section, a stock trading strategy based on GNP with control nodes [17], [89], [90] is used for evaluating the proposed GRA/G portfolio selection method. This is a kind of integrated intelligent model by considering both the portfolio selection and trading strategies. The overall procedure for the GNP approach is shown as Fig. 6.8.

Fig. 6.9 shows a basic structure of GNP. GNP is composed of control nodes, judgment nodes and processing nodes, which are connected to each other. Once GNP is booted up, the execution starts from the control node. The genotype expression of GNP node is also shown in Fig. 6.9. Concretely speaking, K_i represents the node type, and ID_{ip} represents an identification number of the node function at subnode ip. a_{ip} is a parameter which represents the threshold for determining buying or selling stocks in a processing node. Q_{ip} means Q value which is assigned to each state and action pair. C^A_{ip}, C^B_{ip}, ... show the node number of the next node. Judgment node determines the

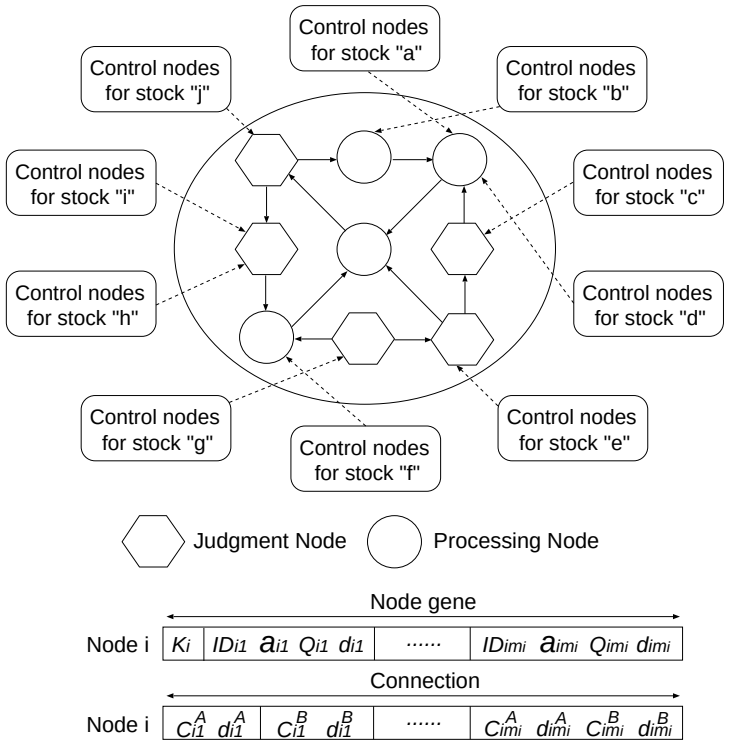


Figure 6.9: Basic structure of GNP

upper suffix of the connection genes depending on the judgment result. d_{ip} is the time delay spent on the judgment or processing, while d_{ip}^{A}, d_{ip}^{B}, ... are time delays spent on the node transition from subnode ip to the next node.

The number of control nodes is fixed and the number of processing nodes activated per control node is evolved. Generally speaking, GNP uses one of the groups of control nodes for one stock brand (as shown in Fig. 6.9), therefore, GNP could deal with multi-brands in the portfolio selected by GRA. When GNP deals with stock "a", it starts the node transition from control node "C_{a1}", and the current activated node returns to one of the control nodes (C_{a1}, ..., C_{an}) successively after transiting m processing nodes from the last control node. The same as GRA and other evolutionary methods, GNP has three kinds of genetic operators: selection, crossover and mutation. In GNP, mutation operation could be executed not only on the connections between nodes but also on the node function ID_{ip} and the number of processing nodes m activated per control node.

Moreover, since the stock trading strategy of GNP was described in detail in Chapter 4 and paper [90], we use the same algorithm and simulation settings in this chapter

Table 6.1: Parameter conditions for evolving GRA/G

Number of individuals	100
Mutation individuals	79
Crossover individuals	20
Elite individual	1
Number of generations	300
Number of nodes	10
P_c	0.3
n	2

and will not repeat the procedure again.

6.5 Experimental Results

In order to confirm the effectiveness of GRA/G in the portfolio selection model, we carried out the trading simulations by GNP using the individuals that were obtained in the last generation of GRA/G. The simulation is divided into two stages: one is used for the training of GRA/G and the other is used for the training and testing of GNP.

- Training (GRA/G): January 4, 2001—December 30, 2003
- Training (GNP): January 4, 2001—December 30, 2003
- Testing (GNP): January 5, 2004—December 30, 2004

Generally speaking, the combination of GRA/G and GNP can be described as follows:

(1) Portfolio selection with GRA/G and GNP-based buying/selling strategy are combined, i.e., the portfolio selection with GRA/G is done firstly, then GNP-based stock trading strategy is done to get profits.

(2) After calculating the final profit of each GRA/G individual in the last generation, we will select the one that gains the largest profit as the optimal portfolio, which is the target of the proposed model.

Table 6.2: Average profits [million yen] in the testing period with different n

n	1/3	1/2	1	2	3
Fitness	4.78	4.61	4.97	5.28	5.01

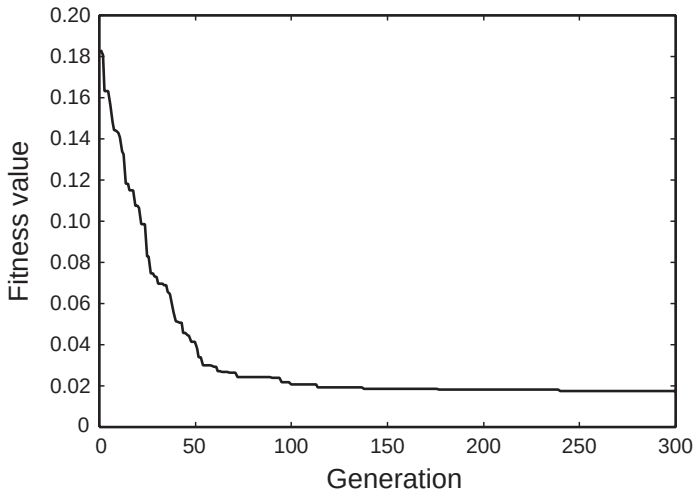


Figure 6.10: Average fitness value in the training period of GRA/G

6.5.1 Performance of GRA/G

Experimental Settings of GRA/G

Table 6.1 shows the parameters of the evolution of GRA/G. The total number of nodes in each individual of GRA/G is 10 which indicates 10 different stocks in a portfolio. Those stocks are selected from the 500 companies listed in the first section of Tokyo stock market in Japan. The content F_i in each node is determined randomly at the beginning of the first generation, and changed appropriately by evolution.

The initial connections between nodes are also determined randomly at the first generation. At the end of each generation, 79 new individuals are produced by mutation, 20 new individuals are produced by crossover, and the best individual is preserved. The other parameters for crossover and mutation are the ones showing good results in the simulations. The terminal condition is 300 generations.

Table 6.3: Stock brands in the optimal portfolio selected by GRA/G and traditional GRA

Stock brands	Optimal portfolio selected by GRA/G	Optimal portfolio selected by GRA
a	Nikkei 300 Stock Index Listed Fund	Nissin Foods Products Co., Ltd.
b	Asahi Kasei Corporation	Hitachi Chemical Co., Ltd.
c	Fujifilm Holdings Corporation	ToTo Ltd.
d	Lion Corporation	Toshiba Corporation
e	NSK Ltd.	Honda Motor Co., Ltd.
f	Casio Computer Co., Ltd.	Yamaha Corporation
g	Isetan Co., Ltd.	Toyota Tsusho Corporation
h	Nishi-Nippon Railroad Co., Ltd.	All Nippon Airways Co., Ltd.
i	Nippon Television Network Corporation	Chubu Electric Power Co., Inc.
j	Shikoku Electric Power Company Incorporated	Toho Co., Ltd.
Profits (yen)	5,275,248 (10.6%)	3,797,128 (7.6%)

Simulation Results of GRA/G

As described in Eq. (2), the value of n in the equation may strongly affect the relation between correlation coefficients ρ and guided mutation rate P_m, thus, we should test various values of n in order to get the optimal portfolio with GRA/G. From Table 6.2, it is clarified that we can get the good profit in the testing period when n is set to 2. Therefore, we set the parameter n=2 in our simulations.

Fig. 6.10 shows the average fitness curve over 30 independent simulations using the data from 2001 to 2003 when the parameter n is set to 2. From Fig. 6.10, we can see that the fitness value is decreasing as the generation goes on.

Moreover, in order to confirm the efficiency of GRA/G, we compared the proposed GRA/G method with the conventional GRA. Through a series of experimental tests, we find that GRA/G generates much higher profit than the traditional GRA model, which means that guided mutation plays an important role as an effective operator in the proposed approach. The stock brands in the optimal portfolio selected by these two methods are also shown in Table 6.3.

6.5.2 Validation by the Stock Trading Model of Genetic Network Programming

Experimental Settings of GNP

Table 6.4 shows the parameters of the evolution of the conventional GNP method which was proposed in our previous study [90]. GNP uses the judgment nodes to judge the information from stock markets, and uses the processing node to take buying and

Table 6.4: Parameter conditions for evolving GNP

Number of individuals	300
Mutation individuals	179
Crossover individuals	120
Elite individual	1
Number of nodes	80
Judgement node	20
Processing node	10
Control node	50
Number of sub-node in each node	2
P_c	0.1
P_m	0.03
α	0.1
γ	0.3
ϵ	0.1

selling actions. Five control nodes are assigned to each brand. The total number of nodes in each individual is 80 including 10 processing nodes, 20 judgment nodes and 50 control nodes. The initial connections between nodes are also determined randomly at the first generation. At the end of each generation, new individuals are produced by selection, crossover and mutation.

In the validation phase by the stock trading model of GNP, we suppose that the initial funds are 50,000,000 Japanese yen in both training and testing periods. Especially, when we use GNP to test the portfolios generated by GRA/G, one GNP individual has 10 groups of control nodes, each of which deals with one brand, so one GNP can deal with 10 brands in the portfolio.

Simulation Results of GNP

Fig. 6.11 shows the fitness and profit curves, i.e., $Fitness(n)$ and $Profit(t, b, n)$ of each brand $b \in B$ in the nth generation using the data from 2001 to 2003, and the line is the average value over 30 independent simulations. From Fig. 6.11, we can see that GNP can obtain larger profits for the training data as the generation goes on.

Fig. 6.12 shows the changes of the profits $Profit(v, b, d)$ of the selected 10 brands in the testing period by GNP trading model, i.e., the best portfolio in the final generation of GRA/G when the value of parameter n is set at 2, where $Profit(v, b, d)$ is the money

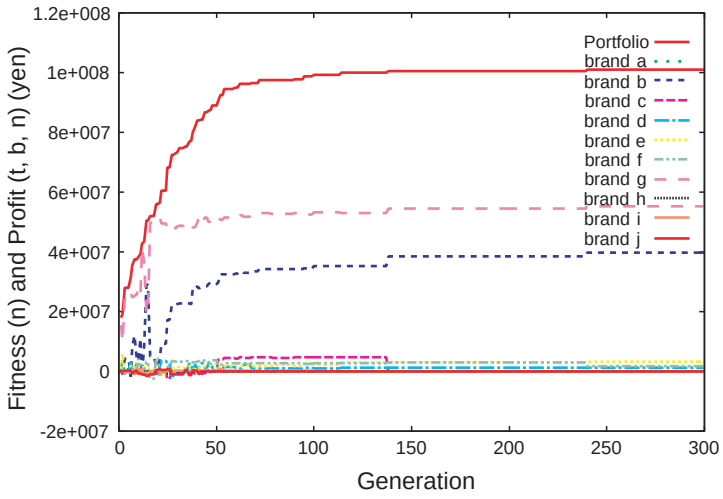


Figure 6.11: Fitness and profit curves of selected 10 brands in the training period by GNP

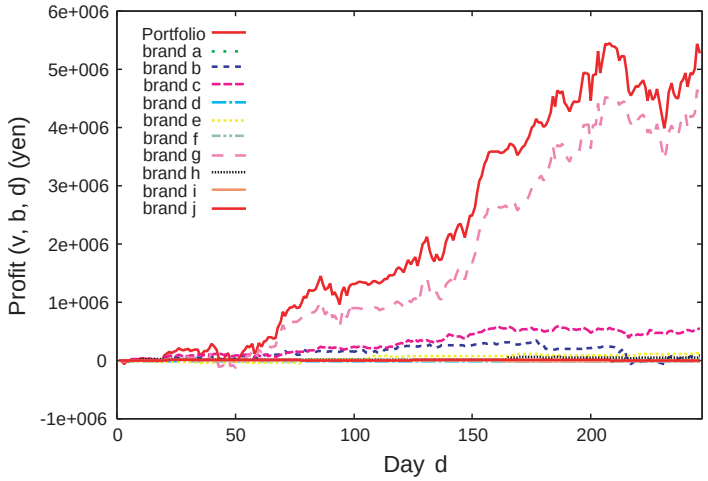


Figure 6.12: Profits change of selected 10 brands in the testing period by GRA/G and GNP

gained or lost during the dealing of brand b until day d for validation. We carried out the dealing of these 10 brands by stock trading model of GNP using the data of 2004. From Fig. 6.12, we can see that the profit keep increasing during the testing period. As a result, by this efficient portfolio optimization system, we can obtain much profits in the trading of those brands.

Table 6.5 shows the profit and profitability in the testing term using the data in 2004. The values in Table 6.5 are the average profit and profitability of 30 independent

Table 6.5: Profits in the testing simulations {Profit[yen](profitability[%])}

Brand	GRA/G&GNP	Buy&Hold
a	-	396,040 (7.9)
b	-	-669,492 (13.4)
c	-	360,231 (7.2)
d	-	278,261 (5.6)
e	-	1,349,010 (27.0)
f	-	1,860,870 (37.2)
g	-	55,038 (1.1)
h	-	30,303 (0.6)
i	-	-259,419 (-5.2)
j	-	165,375 (3.3)
Average	5,275,248 (10.6)	3,566,217 (7.1)

simulations with different random seeds. For the comparison, Table 6.5 also shows the results of Buy&Hold method which is often considered as a benchmark in the stock trading. Buy&Hold buys as much stocks as possible at the opening of the market on the first day in the simulations, and sells all the stocks at the opening on the last day. From Table 6.5, the proposed method can obtain larger profits than Buy&Hold in the trade of 10 brands. Especially, the profit for each brand can be calculated in the Buy&Hold method, however, GNP deals with all of the stock brands in the portfolio with the proposed model, therefore, only the total profit is described.

6.6 Summary

This chapter presents an approach to the large-scale portfolio optimization problem using GRA with a new operator, called guided mutation (GRA/G). Guided mutation generates offspring according to the average value of the correlation coefficients in each individual of GRA. Besides the GRA/G, we adopt a GNP-based strategy for stock trading to get profits. In order to pick up the most efficient portfolio, the algorithm evaluates the relationships between stock brands using a specific measure of strength and generates the optimal portfolio in the final generation. We carried out the experiments using stock data of selected 10 brands for 4 years. In the experiments, the efficiency of GRA/G method is confirmed by the stock trading model of GNP. Compared with conventional methods, the proposed intelligent model has three advantages: first, GRA/G

method considers the correlation coefficient as the strength between stock brands to optimize the portfolio. From the results, it is clarified that we can obtain much profits in the trading of those brands. Second, GNP can adjust the distribution of the initial budget to each selected brand in the portfolio at each generation, and more budget is assigned to the brands with larger profitability. It is different from the Markowitz method in terms of that the distribution ratio of the capital for each brand is determined to maximize the total profits, while the risk factor is already considered in GRA/G. Third, guided mutation is proposed to generate offspring according to the average value of correlation coefficients in each individual. New solutions are generated by applying the guided mutation to the current population, and the resultant solutions will be not far from the best portfolio and fall into a promising area.

The contribution of this chapter showed the efficiency of GRA/G to solve the portfolio optimization problems. Combined with the GNP-based trading strategy, the GRA/G method developed in the chapter can provide an efficient and convenient tool for investors. In the future, the proposed system can be further investigated by incorporating other soft computing techniques or by providing a better model. They include, but not restricted to:

(1) There are numerous models for portfolio selection and stock trading in the literatures. It is important to study the behavior of these models when applied to stock market. Also, we can compare the results of the proposed method with the previous heuristic and mathematical programming algorithms.

(2) The intelligent model presented can be further improved by modifying the fitness function and combining with other approaches.

Chapter 7

Concluding Remarks

The thesis "Study on Stock Trading and Portfolio Optimization using Genetic Network Programming" provides a series of unique techniques on the decision support for the investors in stock market. The contribution of this thesis is to bring forward innovative Genetic Network Programming (GNP) with application to implement strategies of stock trading and portfolio investment. The results and future work are summarized as below.

7.1 Results

The proposed research focuses on the problem of Investment Strategy Determination and Portfolio Optimization through the use of Genetic Network Programming with reinforcement learning technique. The objective of this work is to provide unique techniques of decision-making for investors. First, it presents a stock trading model based on GNP and Sarsa learning by the use of Importance Index (IMX) and candlestick charts. Appropriate trading actions can be determined with the proposed model depending on the changing situations. Second, it provides an application of GNP with control node (GNPcn) to the portfolio optimization problem. Finally, it proposes a new method name Genetic Relation Algorithm (GRA) and applies it to the large-scale portfolio selection problem.

In this study, three advances were made including the efficient stock trading rules, multi brands optimization and portfolio selection. The main realized results are presented as below.

1. Efficient stock trading rules using Genetic Network Programming with reinforcement learning

It has been clarified that GNP is an effective method mainly for dynamic problems such as creating stock trading rules. When GNP is combined with Sarsa learning, an effective stock trading model can be made and it has two advantages: one of them is online learning, and another advantage is the combination of a diversified search of GNP and an intensified search of Sarsa learning. Moreover, Importance Index and candlestick charts are introduced in this model for stock trading decision making. The simulation results show that GNP-Sarsa has obtained good profits and outperforms many other tradition methods in this field, i.e., traditional GNP and benchmark of Buy-and-Hold (Chapter 2).

When GNP-Sarsa model is combined with sliding windows, we can get much better performance due to the advantage of real time updating, which means GNP can adapt to the changing trend of stock prices. Also, as another new point, the function of Importance Index has been optimized by the evolution in order to get more effective judgment functions. The experimental results show that real time updating GNP can get more profits than traditional methods, while the period of sliding window is set as 7 days (Chapter 3).

2. A portfolio optimization model using Genetic Network Programming with control nodes

Since the previous GNP-Sarsa model can only deal with individual stock brand, a new method of GNP with control nodes (GNPcn) was proposed in this chapter, which can deal with multi brands in a portfolio simultaneously. GNPcn has improved the performance of GNP by extending the evolutionary method of it, i.e., the breadth and depth of searching space for GNP. The main problem that we have solved in this part is how to allocate the available capital to different stock brands in order to maximize the profit. From the experimental results, it is clear that the brand which can obtain larger profitability can also get the initial budget more than other brands. The optimization efficiency of the proposed portfolio model has been shown consequently (Chapter 4).

3. A portfolio selection model using Genetic Relation Algorithm and Genetic Network Programming

The final part presents a new approach named Genetic Relation Algorithm (GRA), which is designed for the portfolio selection. Given the investor's objectives and economic conditions, we can find out what stock brands to include in an optimal portfolio by using the proposed GRA method, and thus maximize the expected return and minimize risk simultaneously. In order to pick up the most efficient portfolio from a large number of brands, GRA considers the correlation coefficient between stock brands as strength, which indicates the relation between nodes in GRA individuals. The algorithm evaluates the relationships between stock brands using a specific measure of strength and generates the optimal portfolio in the final generation. The performance of the portfolio selected with GRA is confirmed by the GNPcn stock trading model (Chapter 5).

Moreover, in order to improve the performance of GRA method, a new genetic operator named guided mutation has been developed in the final chapter. With the use of guided mutation, the resultant solution can hopefully fall in or close to a promising area. Therefore, the exploitation ability of evolution has been improved. By carrying out the simulations and comparisons, GRA with guided mutation (GRA/G) shows its effectiveness for portfolio selection (Chapter 6).

7.2 Future Work

Although the stock trading rules and portfolio optimization model based on GNP in this thesis have presented promising results in the realization of investment strategies, there are still some points deserve further investigation.

Firstly, since GNP-Sarsa stock trading model only consider how to maximize the final profits, in the real-world application to the stock market using GNP, we should revise the fitness functions with consideration of transaction costs. In order to improve the performance of the proposed method, we should develop a new method that can learn appropriate calculation periods.

Secondly, it is necessary to create more efficient judgment functions to judge current stock price appropriately in the changing environment, which deserves more explorations.

Finally, the algorithm presented can be further improved by incorporating several new methods to allow the evolution of better GRA individuals, such as modifying the genetic operators of crossover and mutation. Also, the portfolio selection model can be enhanced by modifying the fitness function, and it is necessary to study various measures in addition to the correlation coefficient.

In the future study, the system will be tested on many kinds of markets and with different data periods. Also, we will compare the results of the proposed methods with the previous heuristic and mathematical programming algorithms.

Appendix A

A General Introduction of Genetic Network Programming

A.1 Basic Structure of GNP

A.1.1 Components

Fig. A.1 shows a basic structure of GNP. GNP program consists of one start node, plural judgment nodes and processing nodes which are connected to each other. The role of the start node is to determine the first node to be executed and it has no function and conditional branch. Judgment nodes have conditional branch for decision functions. Each judgment node returns a judgment result and determines the next node to be executed. Processing nodes work as action/processing functions. For example, processing nodes determine the agent's actions such as buying stocks or selling stocks in the stock market. In contrast to judgment nodes, processing nodes have no conditional branch. By separating processing and judgment functions, GNP can handle various combinations of judgment and processing nodes. That is, the number of judgment nodes and the kind of judgment nodes that should be used can be determined by evolution. Suppose there are eight kinds of judgment nodes $(J_1, ..., J_8)$, and four kinds of processing nodes $(P_1, ..., P_4)$. Then, GNP can make a node transition by selecting necessary nodes, e.g., $J_1 \rightarrow J_5 \rightarrow J_3 \rightarrow P_1$. Here, it says that judgment nodes J_1, J_5 and J_3 are needed for processing node P_1. By selecting necessary nodes, GNP program can be quite compact and evolved efficiently.

In my research, as described above, each processing node determines an agent's action such as "buying stocks", "selling stocks" and so on. And each judgment node determines the next node after judging "what kind of technical index should be used?",

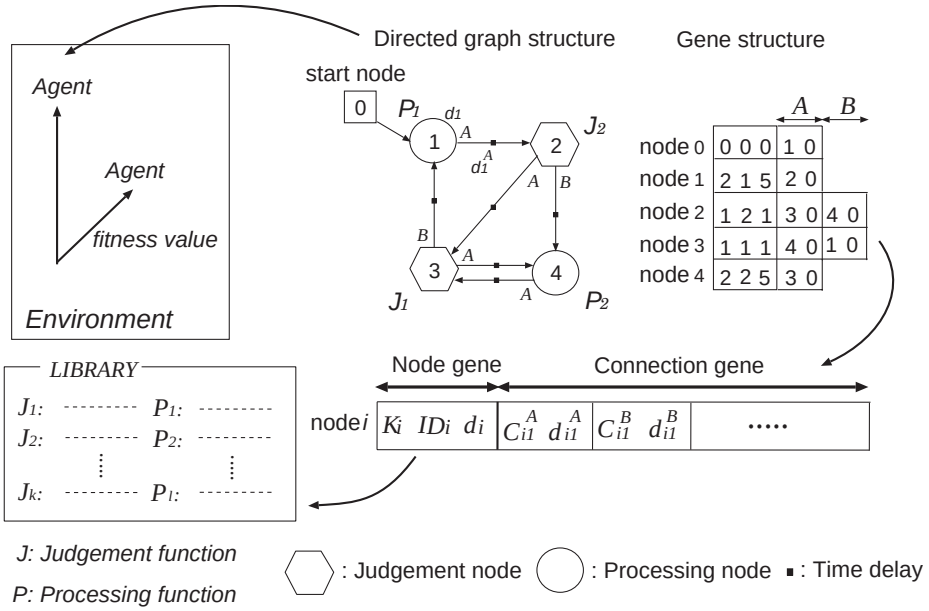


Figure A.1: Basic structure of GNP

"what type of candlestick chart should be used?" and so on. However, in other applications, they could also be applied to other functions such as "judge sensor values (judgment)", "determine wheel speed (processing)" of Khepera robot (by K-Team Corp.), "judge what is in the forward and in the right (judgment)" and "determine go forward or turn right (processing)" in the tileworld [100].

GNP evolves the graph structure with the predefined number of nodes, so it never cause bloating problems that always happen in GP. In addition, GNP has an ability to use certain judgment/processing nodes repeatedly to achieve a task. Therefore, even if the number of nodes is predefined and small, GNP can perform well by making effective node connections based on re-using nodes and memory function. As a result, we do not have to prepare an excessive number of nodes. The compact structure of GNP is quite important and distinguished because of its contribution to saving memory consumption and calculation time.

A.1.2 Memory Function

The node transition begins from a start node while there is no terminal node. The current node is transferred after the start node according to the node connections and judgment results, in other words, the selection of the current node is influenced by

the node transitions of the past. Therefore, the graph structure itself has an implicit memory function of the past agent actions. Although a judgment node is a conditional branch decision function, the GNP program is not merely the aggregate of if-then rules, because it includes information of judgment and processing in the past. For example, in Fig. A.1, after node 1 (processing node P_1) is executed, the next node becomes node 2 (judgment node J_2). Therefore, when the current node is node 2, we can know the previous processing node was P_1.

The node transition of GNP ends when the end condition is satisfied, e.g., when the time step reaches the preassigned one or the GNP program completes the given task.

A.1.3 Time Delays

GNP has two kinds of time delays: time delay GNP spends on judgment or processing, and one it spends on node transitions. In real world problems, when agents judge environments, prepare for actions and take actions, they need time. For example, when a man is walking and sees a puddle before him, he will avoid it. At that moment, it takes some time to judge the puddle (time delay of judgment), to put judgment into action (time delay of transition from judgment to processing) and to avoid the puddle (time delay of processing). Since time delays are listed in each node gene and are unique attributes of each node, GNP can evolve flexible programs considering time delays. In this paper, time delay of each node transition is set at zero time unit, that of each judgment node is one time unit, that of each processing node is five time units, and that of a start node is zero time unit. In addition, the one step of an agent's behavior is defined in such a way that one step ends when an agent uses five or more time units. Thus, an agent should do fewer than five judgments and one processing, or five judgments in one step. Suppose that there are three agents (agent 0, agent 1, agent 2) in an environment. During one step, first agent 0 takes an action, next agent 1, finally agent 2. In this way, agents repeatedly take actions until reaching the maximum preassigned steps. Another important role of time delays and steps is to prevent the program from falling into deadlocks. For example, if an agent cannot execute processing because of the judgment loop, then one step ends after five judgments. Such a program is removed from the population in the evolutionary process, or the node transition is changed by the learning process of GNP-RL, as described later.

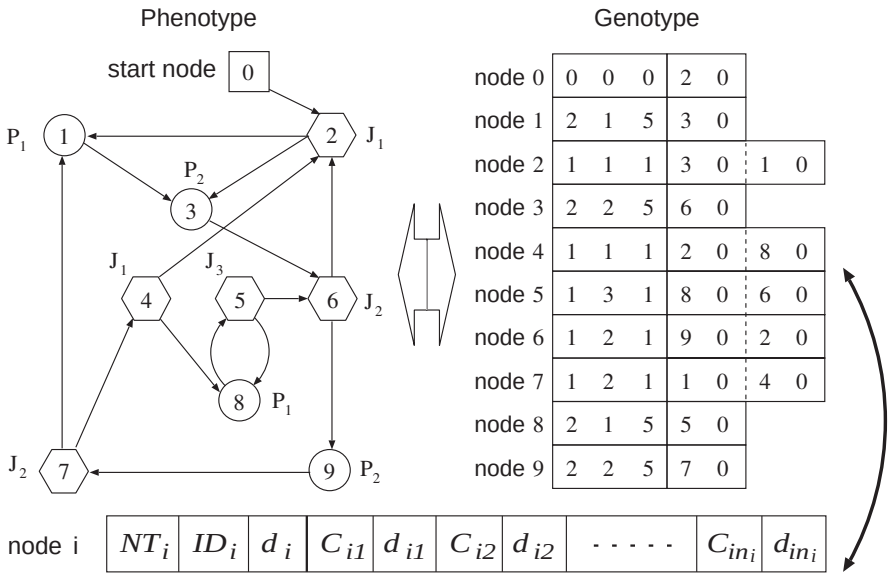


Figure A.2: The genotype and phenotype expression of GNP node

A.2 Genotype and Phenotype of GNP

The graph structure of GNP is determined by the combination of the node genes. So, it is necessary to encode genetic information properly of each node in order to express GNP on a computer. As shown in Fig. A.2, it is the translation from the phenotype to the genotype. Phenotype shows the network structure while the genotype carries out the programming for the calculation on a computer. The genetic information expressing node kinds and connections are written in the genotype, and then we can define the gene set as a GNP program.

The genotype expression of GNP node is shown in Fig. A.2. This describes the gene of node i, then the set of these genes represents the genotype of GNP individuals. All variables in these genes are described by integer. NT_i represents the node type, NT_i=0 means the node i is start node. NT_i=1 means judgment node and NT_i=2 means processing node. ID_i represents the identification number of the node function, e.g., NT_i=1 and ID_i=2 means the node is J_2. C_{i1}, C_{i2}, ... show the node number connected from node i. The total number of connection genes depends on the function of nodes. d_{i1}, d_{i2}, ... mean time delays spent on the transition from node i to node C_{i1}, C_{i2}, ..., respectively. Judgment nodes determine the upper suffix of the connection genes to refer to depending on their judgment results. For example, if the judgment result is

"2", GNP refers to C_{i2} and d_{i2}. However, a start node and processing nodes have no conditional branch.

A.3 Initialization of a GNP population

An initial population is produced according to the following rules. First, we determine the number of each kind of node, therefore all programs in a population have the same number of nodes and the nodes with the same node number have the same function. However, the extended algorithm, GNP with reinforcement learning, determines the node functions automatically, so we only need to determine the number of judgment nodes and processing nodes, e.g., 40 judgment nodes and 20 processing nodes. The connection genes C_{i1}, C_{i2}, ... are set at the values selected randomly from 1, ..., n-1 (except i in order to avoid self-loop).

A.4 A Run of a GNP Program

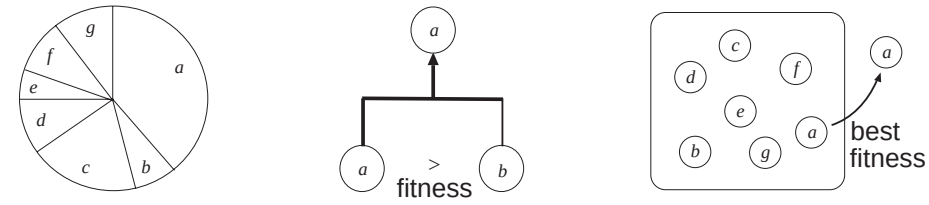
The node transition of GNP is based on C_{ini}. If the current node i is a judgment node, GNP executes the judgment function IDi and determines the next node using its result. If the current node i is a processing node, GNP executes the processing function IDi and transits to the next node.

A.5 Genetic Operators of GNP

Although GNP is a technique discovering optimum solutions by evolving a program, we use a technique based on the genetic operation of GA. Selection, mutation, and crossover established by the genetic operation of GNP are reviewed. In each generation, the elite individuals are preserved and the rest of the individuals are replaced with the new ones generated by crossover and mutation.

A.5.1 Selection

The individuals carrying out crossover or mutation and individuals preserved to the next generation are selected according to the rules based on the fitness of each individual. The following selections are established in GNP as shown in Fig. A.3.



(a) Roulette Selection (b) Tournament Selection (c) Elite Selection

Figure A.3: Selection examples of GNP

- Roulette Selection

  As the roulette selection of GNP, the selection is carried out by the probability in proportion to the relative value of the fitness of the individual. The roulette selection has not been used in conventional research because it could not be applied to the case where lower fitness is dominant.

- Tournament Selection

  Comparing with N individuals selected from the population randomly, the individual having the highest fitness is selected among them. N is the tournament size and N=2 is used generally. The tournament selection is mainly used in conventional GNP research, because the tournament selection is available in the case where lower fitness is dominant.

- Elite Selection

  Comparing with the fitness of all individuals of the population, the elite selection moves M individuals having higher fitness to the next generation. M is the number of elite individual, and the solution converges quickly if M is high.

A.5.2 Crossover

The crossover of GNP based on GA is the genetic operation generating two new offspring by exchanging the genetic information among the genotype of two parents. Exchanging the genetic information in the genotype is the same as the exchange of the subnetworks of GNP in phenotype. The following crossovers are established in GNP.

- One Point Crossover

  Selecting one node as the crossover point randomly, the whole genetic information of its node is exchanged. The example of one point crossover is shown in

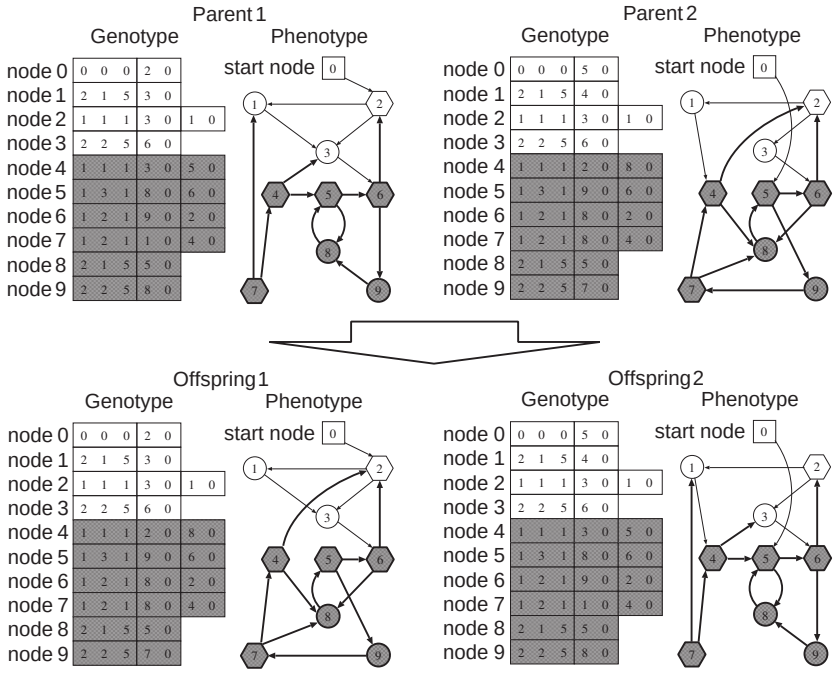


Figure A.4: One point crossover

Fig. A.4. The performance of the generated offspring individual is influenced by the position of the crossover point.

- Several Points Crossover

  Plural nodes (generally two) are selected as the crossover point randomly, the whole genetic information of their nodes is exchanged. The example of several points crossover is shown in Fig. A.5. Exchanging more small sub networks is available by dividing the network into more blocks.

- Uniform Crossover

  Which nodes to be selected as offspring is decided by the predefined crossover probability Pc for each node of the parent individual, the whole genetic information corresponding to the crossover node is exchanged between the parents. The example of uniform crossover is shown in Fig. A.6.

A.5.3 Mutation

The mutation is carried out by changing genetic information randomly in the mutation of GNP based on GA. The kinds of the mutation are classified according to the changing

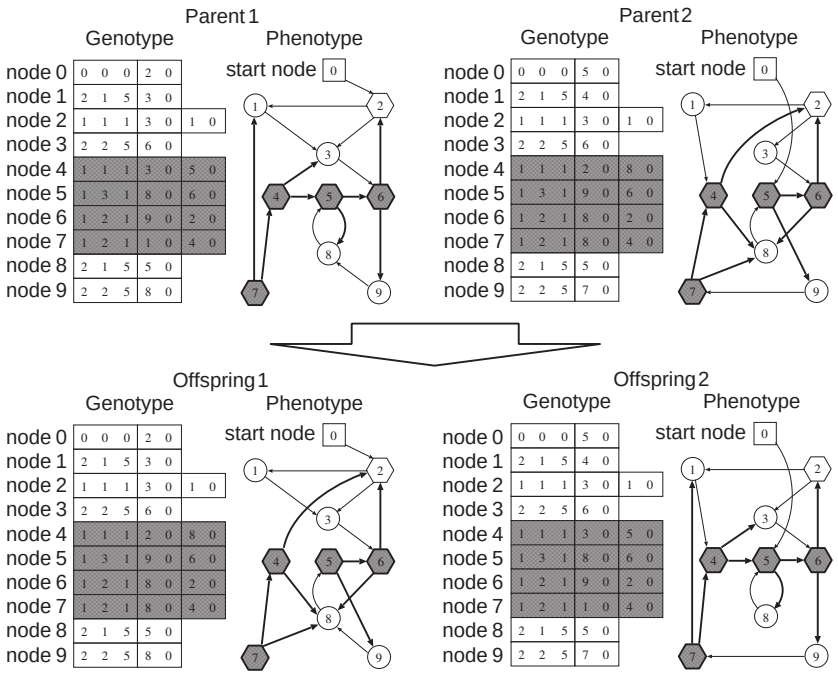


Figure A.5: Several points crossover

information as follows.

- Mutation of connections between nodes

  The connection between nodes is modified. The mutation refers to the change of genetic information C_{ij} of each node in GNP by the predefined probability P_{mc}, and decides the connection for the mutation. Therefore the genetic information C_{ij} is modified in the range of the node number randomly. The example of the mutation is shown in Fig. A.7. In the case of just carrying out this mutation, the kinds of nodes having the same node number are not modified because node type NT_i and identification number ID_i are not modified.

- Mutation of nodes

  The kinds of node are changed. The mutation refers to the change of genetic information NT_i and ID_i of each node in GNP by the predefined probability P_{mn}, and decides the node and the genetic information in mutation. Therefore, the genetic information on the NT_i and ID_i is modified randomly. If the number of the connections are increased by modifying the node function (e.g., modifying the Processing node to the Judgement node), added connections are defined ran-

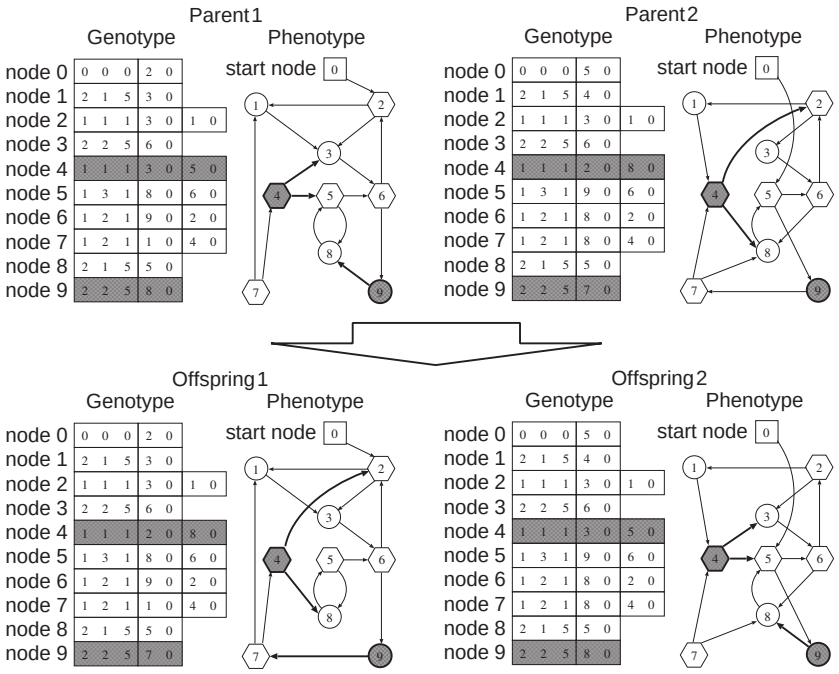


Figure A.6: Uniform crossover

domly. On the other hand, if the number of the connections are decreased, they are deleted. The example of the mutation is shown in Fig. A.8.

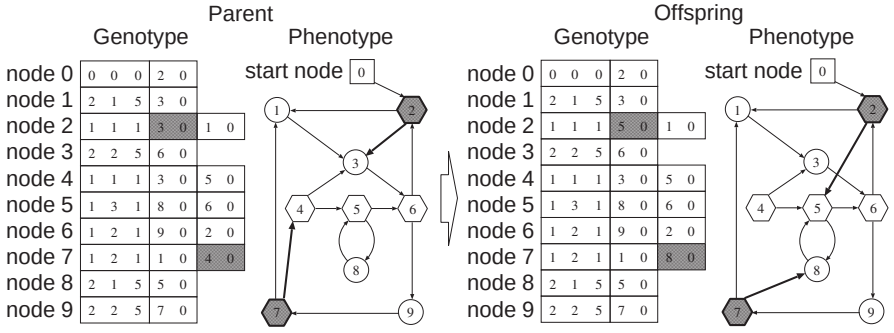


Figure A.7: Connection mutation

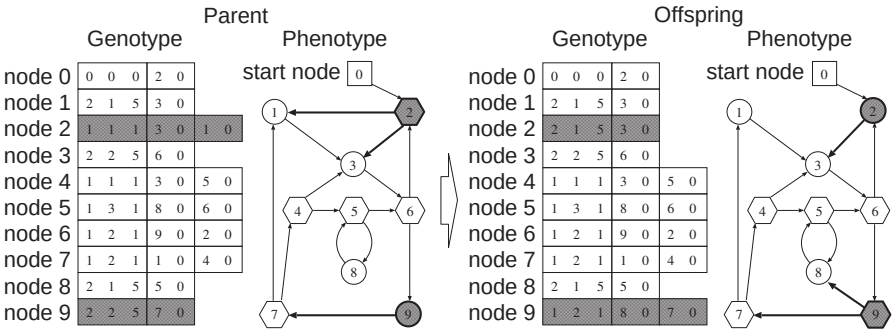


Figure A.8: Node mutation

Appendix B

Technical Indices

In this Appendix, it gives an explanation of the technical indices, which are used as judgment functions in Genetic Network Programming.

- Relative Strength Index (RSI): the sum of the increased price range for a certain period is divided by the sum of the increased and decreased price range for the period.
- Rate of Change (ROC): show the momentum of the stock price and is calculated as follows. Closing price at the current day is divided by the closing price at a certain days before.
- Volume Ratio (VR): the sum of the volume when the stock price rises and the half of the volume when the stock price does not change in a certain period is divided by the sum of the volume in the period.
- Stochastics: judge the position of the current stock price for the highest price and lowest price in a certain period using %K line and %D line. For example, %K of 12 days is calculated as (P-PL)/(PH-PL)*100, and %D of 12 days is calculated as (sum of P-PL for the last three days)/(sum of PH-PL for the last three days)*100. P shows the closing price at the current day, PH shows the highest price in the last 12 days, and PL shows the lowest price in the last 12 days. In this paper, %D line is used to judge the current situation.
- Rank Correlation Index (RCI): give the increase order to the stock prices in a certain period, and show the relation between the order and the number of days.

- Golden/Dead cross (G/D): if the line of the short-term moving average passes through the long-term moving average from the lower side to the upper side, it is called golden cross. If the short-term moving average passes through the long-term one from the upper side to the lower side, it is called dead cross.

- Moving Average Convergence and Divergence (MACD): Generally, the difference between the smoothing average of the stock prices in 12 days and 26 days is called MACD, and the moving average of MACD for nine days is called signal. When MACD passes through signal from the lower side to the upper side, it is judged as the buying sign, and when MACD does from the lower side to the upper side, it is judged as the selling sign.

- Psychological Line (PL): the rate of the days when the stock price increased in a certain period.

- Rate of Deviation from Moving Average (ROD): show how much difference there is between the current stock price and the moving average.

- Rate of Deviation from the current price to the highest price: show how much difference there is between the current stock price and the highest prices in a certain period.

- Rate of Deviation from the lowest price to the current price: show how much difference there is between the current stock price and the lowest price in a certain period.

Acknowledgments

Completing this doctoral work has been a wonderful and overwhelming experience. At the very first, I'm honored to express my deepest gratitude to the most intuitive, smart and supportive advisor, Prof. Kotaro Hirasawa. He has offered me valuable ideas and suggestions with his profound knowledge and research experiences. His patience and kindness are greatly appreciated. I have learned a lot from him not only about dissertation writing, but also the professional ethics. He has fostered certainly the most open, friendly, collaborative and least competitive research group in this Department.

Throughout my three years, I was supported for many semesters by the Japan Society for the Promotion of Science (JSPS), through the generosity of my advisor. Prof. Hirasawa's other students and post-docs, both past and present, comprise a superb research group. The ability to bounce ideas off so many excellent minds has been priceless. My most intense collaboration has been with Dr. Mabu, whose clarity, persistence, ability to create new models, and ability to write new publications, has taught me a lot. It has also been my pleasure to work with Dr. Shimada and Ms. Seguchi. They create such a nice atmosphere for me to work in the lab. My heartiest thanks also go to the former and present members of the Hirasawa Lab for their friendship and assistance over the years. Especially thanks to Prof. Yoshie, Prof. Iwaihara, Prof. Fujimura and Prof. Furuzuki for their invaluable comments and great help during my study career in Japan. There are countless others who have been there for me throughout my time as a graduate student.

My fascination with the academic world is undoubtedly due to the influence of my mother. She taught me a lot at an early age, and let me spend most of my childhood on reading books. My father also gave me great influence in my life, who is highly responsible in our family.

Finally, my husband has been my guiding light and love over these last five years. He has seen my best and my worst, and provided support, hugs, and taken me places I never imagined. Even when my emotional and research brains became so hopelessly, he still stay by my side and give me courage. This work could not be finished without the help and support of many people who are gratefully acknowledged here.

List of Publications

Journals

J1 Y. Chen, S. Mabu and K. Hirasawa, A model of portfolio optimization using time adapting genetic network programming, *Computers & Operations Research*, 11 pages (available online) (2009).

J2 Y. Chen, E. Ohkawa, S. Mabu, K. Shimada and K. Hirasawa, A portfolio optimization model using Genetic Network Programming with control nodes, *Expert Systems with Applications*, Vol. 36, pp. 10735-10745 (2009).

J3 Y. Chen, S. Mabu, K. Shimada and K. Hirasawa, Real Time Updating Genetic Network Programming for Adapting to the Change of Stock Prices, *IEEJ Trans. EIS*, Vol. 129, No. 2, pp. 344-354 (2009).

J4 Y. Chen, S. Mabu, K. Shimada and K. Hirasawa, Trading Rules on Stock Markets Using Genetic Network Programming with Sarsa Learning, *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Vol. 12, No. 4, pp. 383-392 (2008).

J5 S. Mabu, Y. Chen, D. Sohn, K. Shimada and K. Hirasawa, Stock Price Prediction using Neural Networks with RasID-GA, *IEEJ Transactions on Electrical and Electronic Engineering*, Vol. 4, No. 3, pp. 392-403 (2009).

J6 E. Ohkawa, Y. Chen, Z. Bao, S. Mabu, K. Shimada and K. Hirasawa, Buying and Selling Stocks of Multi Brands using Genetic Network Programming with Control Nodes, *IEEJ Trans. EIS*, Vol. 128, No. 12, pp. 1811-1819 (2008).

J7 S. Mabu, Y. Chen and K. Hirasawa, Generating Stock Trading Rules Using Genetic Network Programming with Flag Nodes and Adjustment of Importance Indexes, *IEEJ Trans. EIS*, Vol. 128, No. 9, pp. 1462-1469 (In Japanese) (2008).

J8 S. Mabu, Y. Chen, S. Eto, K. Shimada and K. Hirasawa, Genetic Network Programming with Intron-Like Nodes, *IEEJ Trans. EIS*, Vol. 128, No. 8, pp. 1312-1319 (In Japanese) (2008).

Book Chapters

B1 Y. Chen, S. Mabu and K. Hirasawa, Genetic Network Programming with Reinforcement Learning and Its Application to Creating Stock Trading Rules, pp. 345-360, *Machine Learning*, Published by In-Teh, Vienna, Austria (2009).

International Conferences (with Review Process)

C1 Y. Chen, S. Mabu and K. Hirasawa, A Portfolio Selection Model using Genetic Relation Algorithm and Genetic Network Programming, *In Proc. of the 2009 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 4488-4493, San Antonio, Texas, USA, October, 2009.

C2 Y. Chen, C. Yue, S. Mabu and K. Hirasawa, A Genetic Relation Algorithm with Guided Mutation for the Large-Scale Portfolio Optimization, *In Proc. of the SICE International Annual conference 2009*, pp. 2579-2584, Fukuoka, Japan, August, 2009.

C3 Y. Chen, S. Mabu, E. Ohkawa and K. Hirasawa, Constructing Portfolio Investment Strategy Based on Time Adapting Genetic Network Programming, *In Proc. of the 2009 IEEE Congress on Evolutionary Computation*, pp. 2379-2386, Trondheim, Norway, May, 2009.

C4 Y. Chen, E. Ohkawa, S. Mabu, K. Shimada and K. Hirasawa, A Stock Trading Model for Multi-Brands Optimization Based on Genetic Network Programming with Control Nodes, *In Proc. of the SICE International Annual conference 2008*, pp. 664-669, Chofu, Tokyo, Japan, August, 2008.

C5 Y. Chen, S. Mabu, K. Shimada and K. Hirasawa, Construction of Portfolio Optimization System using Genetic Network Programming with Control Nodes, *In Proc. of the Genetic and Evolutionary Computation Conference 2008*, pp. 1693-1694, Atlanta, Georgia, USA, July 2008.

C6 S. Mabu, Y. Chen, E. Ohkawa and K. Hirasawa, Stock Trading Strategies by Genetic Network Programming with Flag Nodes, *In Proc. of the Genetic and Evolutionary Computation Conference 2008*, pp. 1709-1710, Atlanta, Georgia, USA, July, 2008.

C7 Y. Chen, S. Mabu, K. Shimada and K. Hirasawa, Real Time Updating Genetic Network Programming for Adapting to the Change of Stock Prices, *In Proc. of the IEEE Congress on Evolutionary Computation 2008*, pp. 370-377, Hong Kong, June 2008.

C8 Y. Chen, S. Mabu, K. Hirasawa and J. Hu, Genetic Network Programming with Sarsa Learning and Its Application to Creating Stock Trading Rules, *In Proc. of the IEEE Congress on Evolutionary Computation 2007*, pp. 220-227, Singapore, September 2007.

Bibliography

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press (1975).

[2] D. E. Goldberg, *Genetic Algorithm in search, optimization and machine learning*. Addison-Wesley (1989).

[3] L. J. Fogel, A. J. Owens and M. J. Walsh, *Artificial Intelligence through simulated Evolution*, JohnWiley & Sons (1966).

[4] I. Rechenberg, *Optimieriung technischer Systeme nach Prinzipien der biologischen Evolution*, Frommann-Holzboog, Stuttgart (1973).

[5] H. Schwefel, *Evolution and Optimum Seeking*, JohnWiley & Sons, New York (1995).

[6] J. R. Koza, *Genetic Programming: on the programming of computers by means of natural selection*, Cambridge, Mass.: MIT Press (1992).

[7] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, Cambridge, Mass.: MIT Press (1994).

[8] P. J. Angeline and J. B. Pollack, Evolutionary module acquisition, In D. Fogel and W. Atmar, editors, *In Proc. of the Second Annual Conference on Evolutionary Programming*, pp. 154-163, La Jolla, CA, USA (1993).

[9] D. B. Fogel, An introduction to simulated evolutionary optimization, *IEEE Trans. Neural Networks*, Vol. 5, No. 1, pp. 3-14 (1994).

[10] A. Teller, Evolving programmers: The co-evolution of intelligent recombination operators, In P. J. Angeline and K. E. J. Kinnear, editors, *Advances in Genetic Programming*, chapter 3, pp. 45-68, MIT Press, Cambridge, MA, USA (1996).

[11] A. Teller and M. Veloso, PADO: learning tree-structured algorithm for orchestration into an object recognition system, *Technical report*, Carnegie Mellon University (1995).

[12] A. Teller and M. Veloso, PADO: A new learning architecture for object recognition, In K. Ikeuchi and M. Veloso, editors, *Symbolic Visual Learning*, Oxford University Press (1996).

[13] G. Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, MIT Press, Cambridge, Mass., London, England (1999).

[14] K. Hirasawa, M. Okubo, H. Katagiri, J. Hu and J. Murata, Comparison between genetic network programming (GNP) and genetic programming (GP), *In Proc. of Congress on Evolutionary Computation*, pp. 1276-1282 (2001).

[15] H. Katagiri, K. Hirasawa and J. Hu, Genetic network programming-application to intelligent agents, *In Proc. of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 3829-3834 (2000).

[16] H. Katagiri, K. Hirasawa, J. Hu and J. Murata, Network structure oriented evolutionary model: genetic network programming and its comparison with genetic programming, *In Proc. of the 2001 Genetic and Evolutionary Computation Conference*, Late Breaking Papers, pp. 219-226 (2001).

[17] S. Mabu, K. Hirasawa and J. Hu, A graph-based evolutionary algorithm: Genetic network programming and its extension using reinforcement learning, *Evolutionary Computation*, MIT Press, Vol. 15, No. 3, pp. 369-398 (2007).

[18] S. Mabu, Y. Chen, K. Hirasawa and J. Hu, Stock Trading Rules Using Genetic Network Programming with Actor-Crotic. *In Proc. of the IEEE Congress on Evolutionary Computation 2007*, pp. 508-515 (2007).

[19] T. Eguchi, K. Hirasawa, J. Hu and N. Ota, Study of evolutionary multiagent models based on symbiosis, *IEEE Trans. Syst., Man and Cybern. B*, Vol. 36, No. 1, pp. 179-193 (2006).

[20] S. Luke and L. Spector, Evolving graphs and networks with edge encoding: Preliminary report, In J. R. Koza, editor, *Late Breaking Paper at the Genetic Programming 1996 Conference*, July 28-31, 1996, pp. 117-124, Stanford University, CA, USA (1996).

[21] R. Poli and W. B. Langdon, Genetic programming with one-point crossover, In P. K. Chawdhry, R. Roy and R. K. Pant, editors, *Second On-line World Conference on Soft Computing in Engineering Design and Manufacturing*, pp. 180-189, Springer-Verlag London (1997).

[22] R. Poli and W. B. Langdon, Schema theory for genetic programming with one-point crossover and point mutation, *Evolutionary Computation*, Vol. 6, No. 3, pp. 231-252 (1998).

[23] S. Luke, Genetic programming produced competitive soccer softbot teams for robocup97, *In Proc. of the Third Anuual Genetic Programming Conference* (1998).

[24] H. Iba, Multi-agent reinforcement learning with genetic programming, *In Proc. of the Third Annual Conference of Genetic Programming*, pp. 167-172 (1998).

[25] S. Kamio and H. Iba, Adaptation technique for integrating genetic programming and reinforcement learning for real robots, *IEEE Trans. on Evolutionary Computation*, Vol. 9, No. 3, pp. 318-333 (2005).

[26] R. S. Sutton and A. G. Barto, *Reinforcement Learning-An Introduction*, Cambridge, Massachusetts, London, England: MIT Press (1998).

[27] S. Mabu, K. Hirasawa and J. Hu, Genetic network programming with reinforcement learning and its performance evaluation, *In Proc. of the 2004 Genetic and Evolutionary Computation*, Part II, pp. 710-711, Seattle, WA (2004).

[28] S. Mabu, K. Hirasawa, J. Hu and J. Murata, Online learning of genetic network programming, *In Proc. of the Congress on Evolutionary Computation*, pp. 321-326 (2002).

[29] S. Mabu, K. Hirasawa, J. Hu and J. Murata, Online learning of genetic network programming and its application to prisoner's dilemma game, *Trans. of IEE Japan*, Vol. 123, Part C, No. 3, pp. 535-543 (2003).

[30] P. J. Angeline, An alternate interpretation of the iterated prisoner's dilemma and the evolution of non-mutual cooperation, *In Proc. of the 4th Artificial Life Conference*, pp. 353-358 (1994).

[31] K. L. Downing, Adaptive genetic programming via reinforcement learning, *In Proc. of the 3rd Genetic and Evolutionary Computation Conference*, pp. 19-26 (2001).

[32] R. H. Crites and A. G. Barto, Elevator group control using multiple reinforcement learning agents, *Machine Learning*, Vol. 33, No. 2-3, pp. 235-262 (1998).

[33] N. Baba, N. Inoue and Y. Yanjun, Utilization of soft computing techniques for constructing reliable decision support systems for dealing stocks, *In Proc. of the International Joint Conference on Neural Networks* (2002).

[34] J.-Y. Potvin, P. Soriano and M. Vallee, Generating trading rules on the stock markets with genetic programming, *Computers & Operations Research*, Vol. 31, pp. 1033-1047 (2004).

[35] K. J. Oh, T. Y. Kim, S.-H.Min and H. Y. Lee, Portfolio algorithm based on portfolio beta using genetic algorithm, *Expert Systems with Application*, Vol. 30, pp. 527-534 (2006).

[36] W. Brock, J. Lakonishok and B. LeBaron, Simple Technical Trading Rules and the Stochastic Properties of Stock Returns, *Journal of Finance*, Vol. 47, No. 5, pp. 1731-1764 (1992).

[37] S. Mabu, H.Hatakeyama, M. T. Thu, K. Hirasawa and J. Hu, Genetic Network Programming with Reinforcement Learning and Its Application to Making Mobile Robot Behavior. *IEEJ Trans. EIS*, Vol.126, No.8, pp.1009-1015 (2006).

[38] K. H. Lee and G.S. Jo, Expert system for predicting stock market timing using a candlestick chart, *Expert Systems with Applications*, Vol.16, pp. 357-364 (1999).

[39] Y. Izumi, T. Yamaguchi, S. Mabu, K. Hirasawa and J. Hu, Trading Rules on the Stock Market using Genetic Network Programming with Candlestick Chart, *In Proc. of the 2006 IEEE Congress on Evolutionary Computation*, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, July 16-21, pp. 8531-8536 (2006).

[40] S. Mabu, Y. Izumi, K. Hirasawa and T. Furuzuki, Trading Rules on Stock Markets Using Genetic Network Progamming with Candle Chart, *T. SICE*, Vol.43, No.4, pp. 317-322 (2007) (in Japanese).

[41] Y. Izumi, K. Hirasawa and T. Furuzuki, Trading Rules on the Stock Markets Using Genetic Network Progamming with Importance Index, *T. SICE*, Vol.42, No.5, pp. 559-566 (2006) (in Japanese).

[42] V. Dhar, A Comparison of GLOWER and Other Machine Learning Methods for Investment Decision Making, *EPIA 2001*, Springer Berlin Press, pp. 208-220 (2001).

[43] S. Duerson, F. S. Khan, V. Kovalev and A. H. Malik, Reinforcement Learning in Online Stock Trading Systems (2005).

[44] S. Pafka, M. Potters and I. Kondor, Exponential Weighting and Random-Matrix-Theory-Based Filtering of Financial Covariance Matrices for Portfolio Optimization, Available at http://arxiv.org/pdf/cond-mat/0402573 (2004).

[45] N. Basalto, R. Bellotti, F. De Carlo, P. Facchi and S. Pascazio, Clustering stock market companies via chaotic map synchronization, *Physica A: Statistical Mechanics and its Applications*, Vol. 345, Issue 1-2, pp. 196-206 (2004).

[46] W. Huang, Y. Nakamori and S. Y. Wang, Forecasting stock market movement direction with support vector machine Source, *Computers & Operations Research*, Vol. 32, Issue 10, pp. 2513-2522 (2005).

[47] M. B. Porecha, P. K. Panigrahi, J. C. Parikh, C. M. Kishtawal and S. Basu, Forecasting non-stationary financial time series through genetic algorithm, arXiv:nlin/0507037v1 (2005).

[48] M. H. Jensen, A. Johansen, F. Petroni and I. Simonsen, Inverse Statistics in the Foreign Exchange Market, *Physica A: Statistical Mechanics and its Applications*, Vol. 340, Issue 4, pp. 678-684 (2004).

[49] T. Mikosch and C. Starica, Stock Market Risk-Return Inference. An Unconditional, Non-parametric Approach, Available at SSRN: http://ssrn.com/abstract=882820 (2003).

[50] A. Loraschi, A. Tettamanzi, M. Tomassini, C. Svizzero, C. Scientifico and P. Verda, Distributed Genetic Algorithms With An Application To Portfolio Selection, *In Artificial neural nets and genetic*, pp. 384-387 (1995).

[51] K.-S. Shin and Y.-J. Lee, A genetic algorithm application in bankruptcy prediction modeling, *Expert Systems with Applications*, Vol. 23, No. 3, pp. 321-328 (2002).

[52] S. Mahfoud and G. Mani, Financial forecasting using genetic algorithms, *Journal of Applied Artificial Intelligence*, Vol. 10, No. 6, pp. 543-565 (1996).

[53] G. G. Szpiro, Forecasting chaotic time series with genetic algorithms, *Physical Review E*, Vol. 55, No. 3, pp. 2557-2568 (1997).

[54] A. F. Sheta and J. K. De, Time-series forecasting using GA-tuned radial basis functions, *Information Sciences*, Vol. 133, No. 3, pp. 221-228 (2001).

[55] B. S. Mulloy, R. L. Riolo and R. S. Savit, Dynamics of genetic programming and chaotic time series prediction, *In Proc. of the First Annual Conference on Genetic Programm*, pp. 166-174 (1996).

[56] M. A. H. Dempster and C. M. Jones, A real-time adaptive trading system using genetic programming, *Quantitative Finance*, Vol. 1, pp. 397-413 (2001).

[57] H. Iba and T. Sasaki, Using Genetic Programming to Predict Financial Data, *In Proc. of the 1999 Congress on Evolutionary Computation* (CEC99) (1999).

[58] P. Skolpadungket, K. Dahal and N. Harnpornchai, Profolio Optmization using Multi-objective Genetic Algorithms, *In Proc. of the IEEE Congress on Evolutionary Computation 2007*, pp. 516-523 (2007).

[59] Y. S. Abu-Mostafa, A. F. Atiya, M. Magdon-Ismail and H. White, Introduction to the special issue on neural networks in financial engineering, *IEEE Trans. Neural Netw.*, Vol. 12, No. 4, pp. 653-656 (2001).

[60] Y. Chen, S. Mabu, K. Shimada and K. Hirasawa, Trading Rules on Stock Markets Using Genetic Network Programming with Sarsa Learning, *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Vol. 12, No. 4, pp. 383-392 (2008).

[61] S. Mabu, K. Hirasawa and T. Furuzuki, Trading Rules on Stock Markets Using Genetic Network Programming with Reinforcement Learning and Importance Index, *IEEJ Trans. EIS*, Vol. 127, No. 7, pp. 1061-1067 (2007).

[62] S. Eto, S. Mabu, K. Hirasawa and J. Hu, Genetic Network Programming Considering the Evolution of Breadth and Depth, *In Proc. of the SICE-ICASE International Joint Conference 2006*, pp. 5504-5508 (2006).

[63] E. Ohkawa, Y. Chen, Z. Bao, S. Mabu, K. Shimada and K Hirasawa, Buying and selling Stocks of Multi Brands Using Genetic Network Programming with Control Nodes, *IEEJ Trans. EIS*, Vol. 128, No. 12, pp. 1811-1819 (2008).

[64] H. M. Markowitz, Portfolio selection, *Journal of Finance*, Vol. 7, pp. 77-91 (1952).

[65] H. M. Markowitz, The optimization of a quadratic function subject to linear constraints, *Naval Research Logistics Quarterly*, Vol. 3, pp. 111-133 (1956).

[66] H. M. Markowitz, *Portfolio selection: efficient diversification of investments*, New York: Wiley, Yale University Press (1959).

[67] H. M. Markowitz, *Mean-variance analysis in portfolio choice and capital markets*, Oxford: Blackwell Publishers (1987).

[68] G. E. P. Box and G. M. Jenkins, *Time series analysis: forecasting and control*, San Fransisco, CA: Holden-Day (1976).

[69] H. Tong and K. S. Lim, Threshold autoregressive, limit cycles and cyclical data, *Journal of the Royal Statistical Society Series B*, Vol. 42, No. 3, pp. 245-292 (1980).

[70] N. Sarantis, Nonlinearities, cyclical behavior and predictability in stock markets: international evidence, *International Journal of Forecasting*, Vol. 17, No. 3, pp. 459-482 (2001).

[71] R. F. Engle, Autoregressive conditional heteroskedasticity with estimates of the variance of UK inflation, *Econometrica*, Vol. 50, pp. 987-1008 (1982).

[72] D. Enke and S. Thawornwong, The use of data mining and neural networks for forecasting stock market returns, *Expert Systems with Applications*, Vol. 29, pp. 927-940 (2005).

[73] V. Dropsy, Do macroeconomic factors help in predicting international equity risk premia? *Journal of Applied Business Research*, Vol. 12, pp. 120-132 (1996).

[74] M. Lam, Neural network techniques for financial performance prediction: integrating fundamental and technical analysis, *Decision Support Systems*, Vol. 37, pp. 567-581 (2004).

[75] L. Yu, S. Wang and K. K. Lai, Neural network-based mean-variance-skewness model for portfolio selection, *Computers & Operations Research*, Vol. 35, No. 1, pp. 34-46 (2008).

[76] K.-J. Kim and I. Han, Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index, *Expert Systems with Applications*, Vol. 19, pp. 125-132 (2000).

[77] D. Lin, S. Wang and H. Yan, A multiobjective genetic algorithm for portfolio selection problem, *In Proceedings of the ICOTA*, Hong Kong, December 2001 (2001).

[78] C. C. Lin and Y. T. Liu, Genetic algorithms for portfolio selection problems with minimum transaction lots, *European Journal of Operational Research*, Vol. 185, No. 1, pp. 393-404 (2008).

[79] H. Etemadi, A. A. A. Rostamy and H. F. Dehkordi, A genetic programming model for bankruptcy prediction: Empirical evidence from Iran, *Expert Systems with Applications*, Vol. 36, No. 2, pp. 3199-3207 (2009).

[80] S. Sette and L. Boullart: Genetic programming, principles and applications, *Engineering Applications of Artificial Intelligence*, Vol. 14, No. 6, pp. 727-736 (2001).

[81] J. P. Marney, C. Fyfe, H. Tarbert and D. Miller, Risk adjusted returns to technical trading rules: a genetic programming approach, Computing in Economics and Finance, Society for Computational Economics, Yale University, USA, June, paper No. 147 (2001).

[82] Y. Romahi and Q. Shen, Dynamic financial forecasting with automatically induced fuzzy associations, *In Proceedings of the 9th international conference on fuzzy systems*, pp. 493-498 (2000).

[83] M. J. Best and J. K. Kale, Quadratic programming for large-scale portfolio optimization, In: Keyes J, editor, *Financial services information systems*, Boca Raton: CRC Press LLC, pp. 513-529 (2000).

[84] M. Stein, J. Branke and H. Schmeck, Efficient implementation of an active set algorithm for large-scale portfolio selection, *Computers & Operations Research*, Vol. 35, No. 12, pp. 3945-3961 (2008).

[85] A. Fernandez and S. Gomez, Portfolio selection using neural networks, *Computers & Operations Research*, Vol. 34, No. 4, pp. 1177-1191 (2007).

[86] Y. Xia, B. Liu, S. Wang and K. Lai, A model for portfolio selection with order of expected returns, *Computers & Operations Research*, Vol. 27, No. 5, pp. 409-422 (2000).

[87] K. Hirasawa, T. Eguchi, J. Zhou, L. Yu, J. Hu and S. Markon, A Double-Deck Elevator Group Supervisory Control System Using Genetic Network Programming, *IEEE Transactions on Systems, Man and Cybernetics C*, Vol. 38, No. 4, pp. 535-550 (2008).

[88] E. Gonzales, K. Taboada, K. Shimada, S. Mabu and K. Hirasawa, Evaluating Class Association Rules using Genetic Relation Programming, *In Proc. of the IEEE Congress on Evolutionary Computation 2008*, pp. 731-736 (2008).

[89] Y. Chen, S. Mabu, K. Shimada and K. Hirasawa, Real Time Updating Genetic Network Programming for Adapting to the Change of Stock Prices, *IEEJ Trans. EIS*, Vol. 129, No. 2, pp. 344-354 (2009).

[90] Y. Chen, E. Ohkawa, S. Mabu, K. Shimada and K. Hirasawa, A portfolio optimization model using Genetic Network Programming with control nodes, *Expert Systems with Applications*, Vol. 36, pp. 10735-10745 (2009).

[91] H. Konno and H. Yamazaki, Mean-absolute deviation portfolio in optimization model and its application to Tokyo stock market, *Management Science*, Vol. 37, No. 5, pp. 519-531 (1991).

[92] M. Liu and Y. Gao, An algorithm for portfolio selection in a frictional market, *Applied Mathematics and Computation*, Vol. 182, No. 2, pp. 1629-1638 (2006).

[93] A. Atiya, Bankruptcy prediction for credit risk using neural networks: A survey and new results, *IEEE Transactions on Neural Networks*, Vol. 12, No. 4, pp. 929-935 (2001).

[94] N. Baba, N. Inoue and H. Asakawa, Utilization of neural networks and GAs for constructing reliable decision support systems to deal stocks, *In Proceedings of the IEEE international joint conference on neural networks*, Vol. 3, pp. 111-116 (2000).

[95] S.-C. T. Chou, C.-C. Yang, C.-H. Chan and F. Lai, A rule-based neural stock trading decision support system, *In Proceedings of the IEEE/IAFE 1996 conference on computational intelligence for financial engineering*, pp. 148-154 (1996).

[96] T. Kimoto, K. Asakawa, M. Yoda and M. Takeoka, Stock market prediction system with modular neural networks, *In Proceedings of the IEEE international joint conference on neural networks*, Vol. 1, pp. 1-6 (1990).

[97] P. C. Chang and C. H. Liu, A TSK type fuzzy rule based system for stock price prediction, *Expert Systems with Applications*, Vol. 34, No. 1, pp. 135-144 (2008).

[98] P. C. Chang and Y. W. Wang, Fuzzy Delphi and back-propagation model for sales forecasting in PCB industry, *Expert Systems with Applications*, Vol. 30, No. 4, pp. 715-726 (2006).

[99] F. Glover and M. Laguna, *Tabu Search*, Norwell, MA: Kluwer (1998).

[100] M. E. Pollack and M. Ringuette, Introducing the tile-world: Experimentally evaluating agent architectures, *In Proc. of the conference of the American Association for Artificial Intelligence*, pp. 183-189 (1990).